09/998092

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
1	IS&R	L1	6413	(707/3,104.1).CCLS.	USPAT; US-PGPUB; EPO	2004/05/06 10:48	
2	IS&R	L2	0	("rayyanande-mail").PN.	USPAT; US-PGPUB; EPO	2004/05/06 10:48	
3	BRS	L3	15	rayyan and e-mail	USPAT; US-PGPUB; EPO	2004/05/06 10:51	
4	BRS	L4	1	1 and ((e-mail) same (lexi$4 or syntac$4) same search)	USPAT; US-PGPUB; EPO	2004/05/06 11:09	
5	BRS	L5	936	1 and (e-mail)	USPAT; US-PGPUB; EPO	2004/05/06 10:53	
6	BRS	L6	320	5 and ((search adj input or keyword$3))	USPAT; US-PGPUB; EPO	2004/05/06 10:54	
7	BRS	L7	24	6 and (media adj content or (webpages))	USPAT; US-PGPUB; EPO	2004/05/06 11:03	
8	BRS	L8	0	e-mail.ti. and serach.ti.	USPAT; US-PGPUB; EPO	2004/05/06 11:03	
9	BRS	L9	1	e-mail.ti. and search.ti.	USPAT; US-PGPUB; EPO	2004/05/06 11:03	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
10	BRS	L10	1	(media adj agent).ti.	USPAT; US-PGPUB; EPO	2004/05/06 11:04	
11	BRS	L11	5137	1 and @ad<20011130	USPAT; US-PGPUB; EPO	2004/05/06 11:16	
12	BRS	L12	65	11 and (media adj content)	USPAT; US-PGPUB; EPO	2004/05/06 11:09	
13	BRS	L13	6	12 and ((lexi$4 or syntac$4) same search)	USPAT; US-PGPUB; EPO	2004/05/06 11:15	
14	BRS	L14	150619	analyzing naer2 e-mail	USPAT; US-PGPUB; EPO	2004/05/06 11:15	
15	BRS	L15	46	analyzing near2 e-mail	USPAT; US-PGPUB; EPO	2004/05/06 11:36	
16	BRS	L16	22	15 and @ad<20011130	USPAT; US-PGPUB; EPO	2004/05/06 11:47	
17	IS&R	L17	1	("6115709").PN.	USPAT; US-PGPUB; EPO	2004/05/06 11:24	
18	BRS	L18	0	analyzing near2 (word adj processor adj document)	USPAT; US-PGPUB; EPO	2004/05/06 11:37	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
19	BRS	L19	1	analyzing same (word adj processor adj document)	USPAT; US-PGPUB; EPO	2004/05/06 11:39	
20	BRS	L20	0	11 and (parsing near2 e-mail)	USPAT; US-PGPUB; EPO	2004/05/06 11:39	
21	BRS	L21	11	11 and (parsing same e-mail)	USPAT; US-PGPUB; EPO	2004/05/06 11:47	
22	BRS	L22	1	"6081773".PN.	USPAT	2004/05/06 11:43	
23	BRS	L23	1	"6076088".PN.	USPAT	2004/05/06 11:43	
24	BRS	L24	1	"6070133".PN.	USPAT	2004/05/06 11:43	
25	BRS	L25	1	"5991710".PN.	USPAT	2004/05/06 11:43	
26	BRS	L26	1	"4965763".PN.	USPAT	2004/05/06 11:43	
27	BRS	L27	3	704/1,4	USPAT; US-PGPUB; EPO	2004/05/06 11:47	
28	IS&R	L28	459	(704/1,4).CCLS.	USPAT; US-PGPUB; EPO	2004/05/06 11:47	
29	BRS	L29	378	28 and @ad<20011130	USPAT; US-PGPUB; EPO	2004/05/06 11:47	
30	BRS	L30	31	29 and (pars$4 same (e-mail or (document)))	USPAT; US-PGPUB; EPO	2004/05/06 11:50	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
31	BRS	L31	3	29 and (pars$4 same (e-mail or (word adj processor adj document)))	USPAT; US-PGPUB; EPO	2004/05/06 11:48	
32	BRS	L32	28	30 not 31	USPAT; US-PGPUB; EPO	2004/05/06 11:51	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
31	BRS	L31	3	29 and (pars$4 same (e-mail or (word adj processor adj document)))	USPAT; US-PGPUB; EPO	2004/05/06 11:48	
32	BRS	L32	28	30 not 31	USPAT; US-PGPUB; EPO	2004/05/06 11:51	
33	IS&R	L33	1487	(707/4).CCLS.	USPAT; US-PGPUB; EPO	2004/05/06 13:48	
34	BRS	L34	1417	33 and @ad<20011130	USPAT; US-PGPUB; EPO	2004/05/06 13:48	
35	BRS	L35	43	34 and (prediction)	USPAT; US-PGPUB; EPO	2004/05/06 13:49	
36	BRS	L36	21	35 and (e-mail or word or word adj perfect)	USPAT; US-PGPUB; EPO	2004/05/06 13:56	
37	IS&R	L37	15	(("6510406") or ("6282549") or ("6347313") or ("5873056") or ("5889506") or ("5619709") or ("6134532") or ("5855015") or ("6189002") or ("6038560") or ("6304864") or ("6094652") or ("6311194") or ("5987457") or ("6175829")).PN.	USPAT; US-PGPUB; EPO	2004/05/06 13:59	

US 20020038299A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0038299 A1

Zernik et al. (43) Pub. Date: Mar. 28, 2002

(54) INTERFACE FOR PRESENTING INFORMATION

(76) Inventors: Uri Zernik, Palo Alto, CA (US); Dror Zernik, Haifa (IL); Doron Myersdorf, Foster City, CA (US)

Correspondence Address:
RITTER VAN PELT & YI, L.L.P.
4906 EL CAMINO REAL
SUITE 205
LOS ALTOS, CA 94022

(21) Appl. No.: 09/764,336

(22) Filed: Jan. 16, 2001

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/190,848, filed on Mar. 20, 2000.

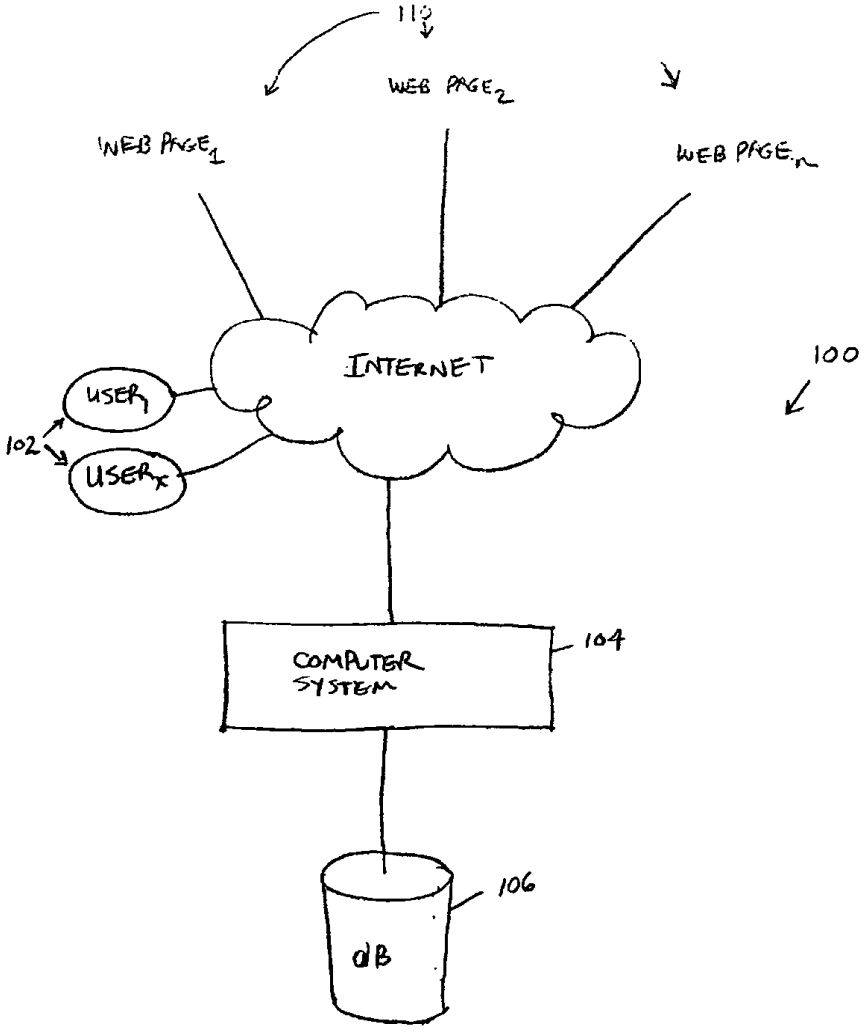
Publication Classification

(51) Int. Cl.[7] G06F 7/00

(52) U.S. Cl. 707/3

(57) ABSTRACT

A system and method are disclosed for presenting information. Categories are determined for found information by analyzing the content of the information. The categories are correlated with images that represent the categories. Images are displayed that correspond to the categories.



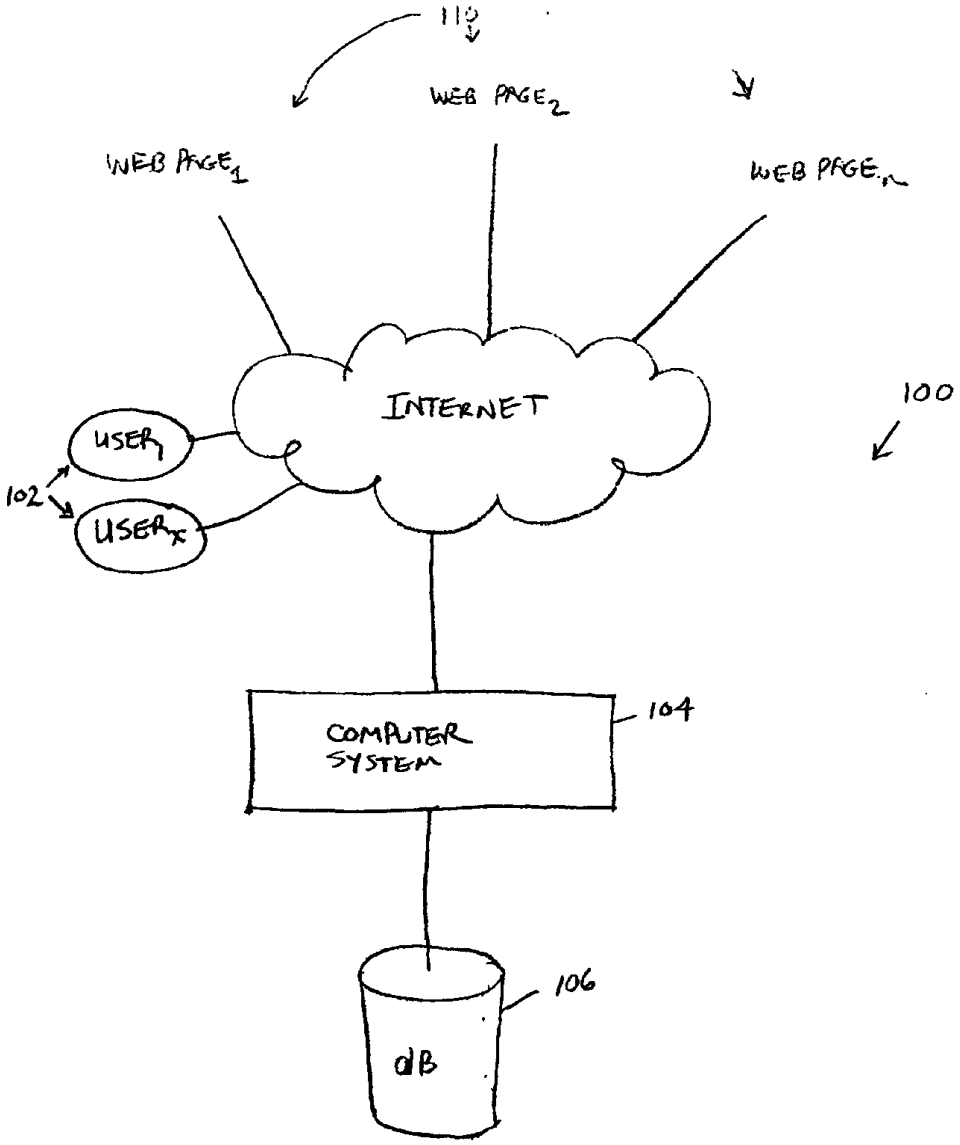

112
WEB PAGE2
WEB PAGE1
WEB PAGEn
INTERNET
100
USER1
102
USERx
COMPUTER SYSTEM
104
106
dB

FIG. 1

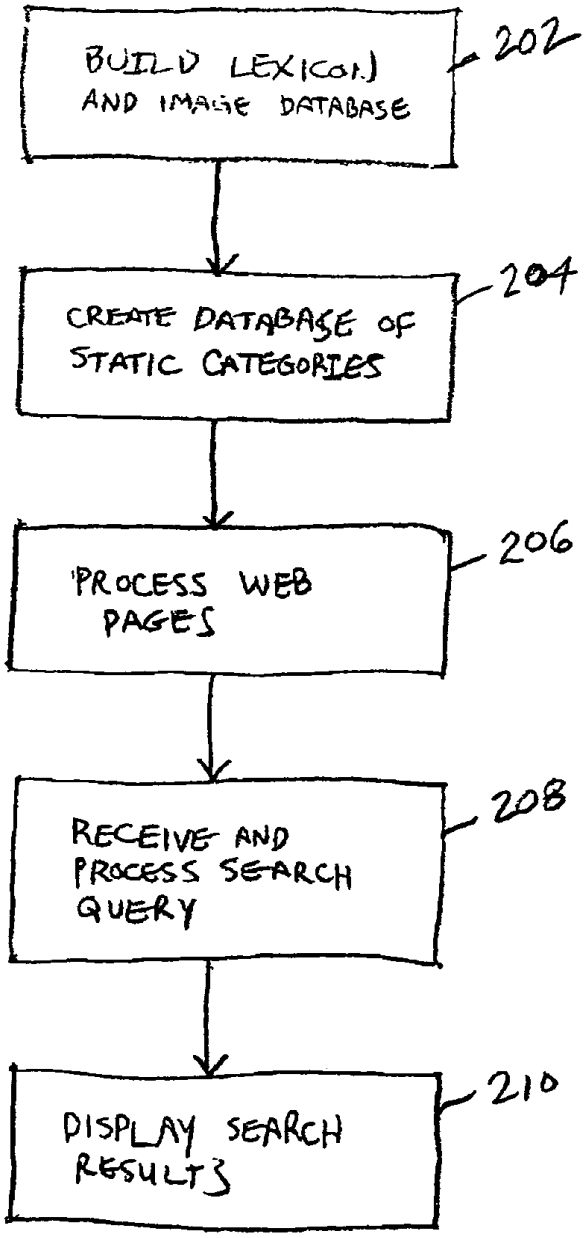

BUILD LEXICON AND IMAGE DATABASE
202
CREATE DATABASE OF STATIC CATEGORIES
204
PROCESS WEB PAGES
206
RECEIVE AND PROCESS SEARCH QUERY
208
DISPLAY SEARCH RESULTS
210

FIG. 2

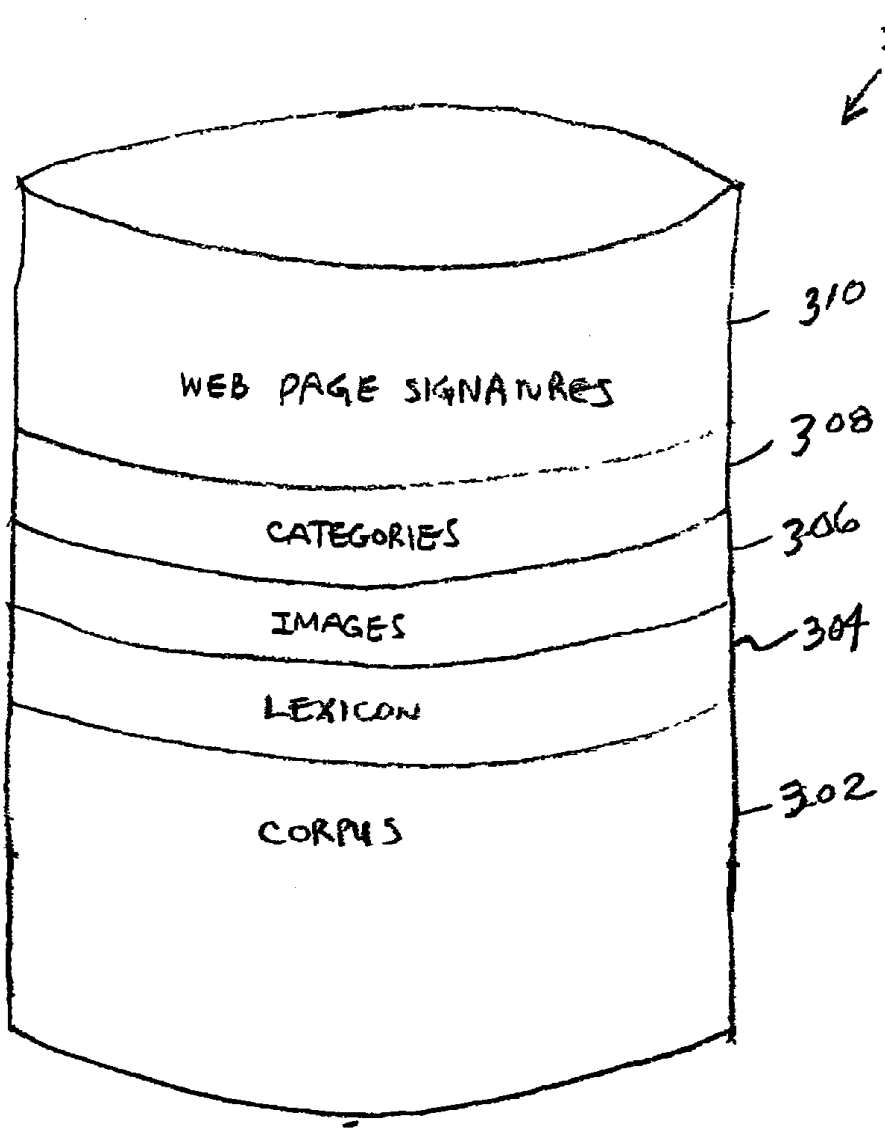

300
WEB PAGE SIGNATURES
310
308
CATEGORIES
306
IMAGES
304
LEXICON
CORPUS
302

FIG. 3

CREATE DATABASE OF CATEGORIES

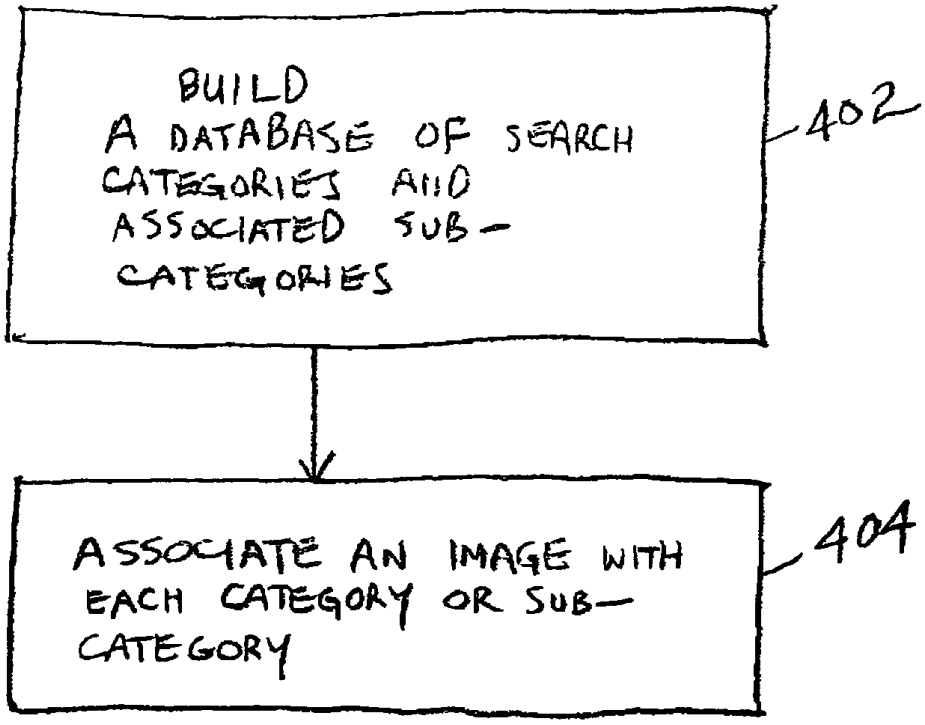


FIG. 4

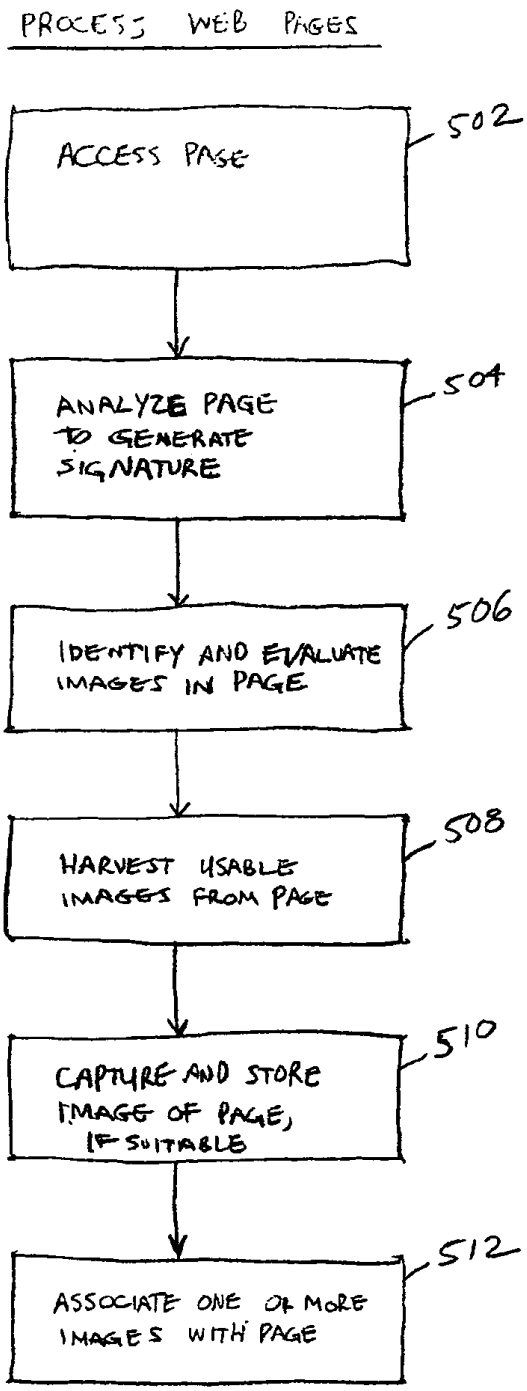

PROCESS WEB PAGES
ACCESS PAGE
502
ANALYZE PAGE TO GENERATE SIGNATURE
504
IDENTIFY AND EVALUATE IMAGES IN PAGE
506
HARVEST USABLE IMAGES FROM PAGE
508
CAPTURE AND STORE IMAGE OF PAGE, IF SUITABLE
510
ASSOCIATE ONE OR MORE IMAGES WITH PAGE
512

FIG. 5

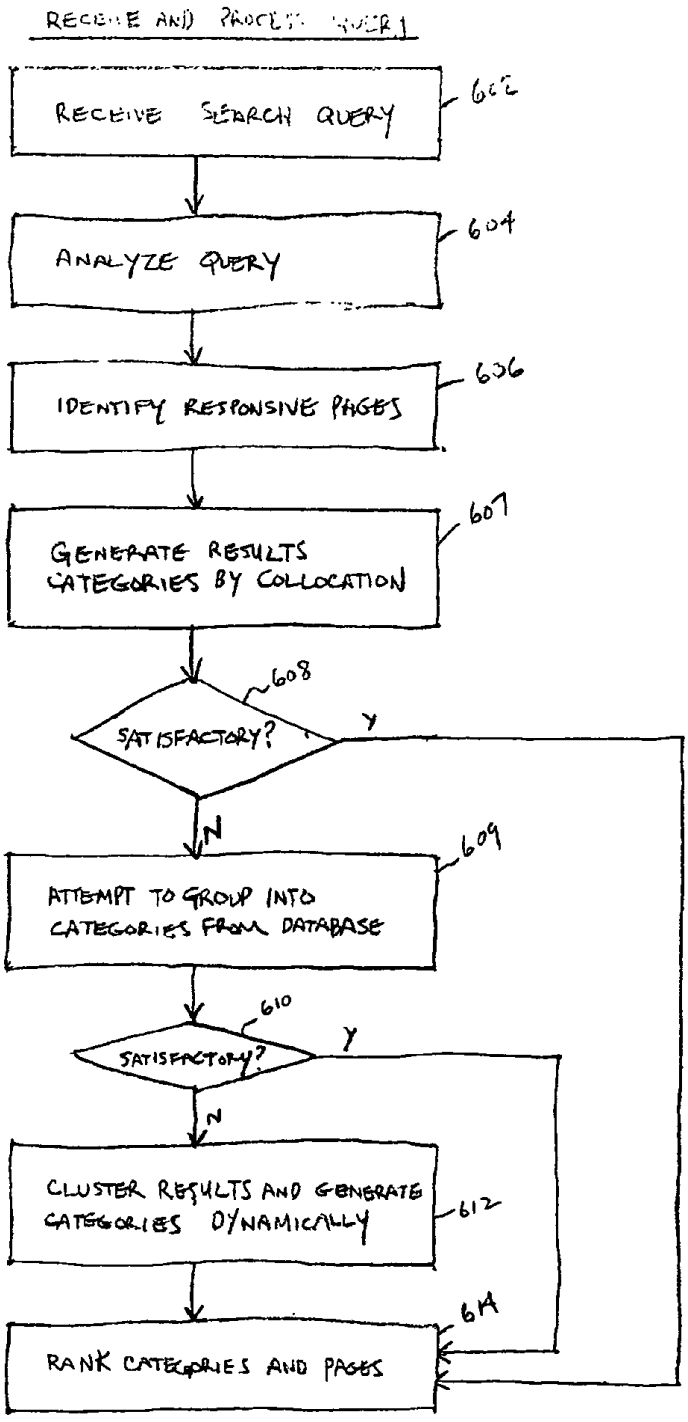

RECEIVE AND PROCESS QUERY
RECEIVE SEARCH QUERY
602
ANALYZE QUERY
604
IDENTIFY RESPONSIVE PAGES
606
GENERATE RESULTS CATEGORIES BY COLLOCATION
607
SATISFACTORY?
608
Y
N
ATTEMPT TO GROUP INTO CATEGORIES FROM DATABASE
609
SATISFACTORY?
610
Y
N
CLUSTER RESULTS AND GENERATE CATEGORIES DYNAMICALLY
612
RANK CATEGORIES AND PAGES
614

FIG. 6

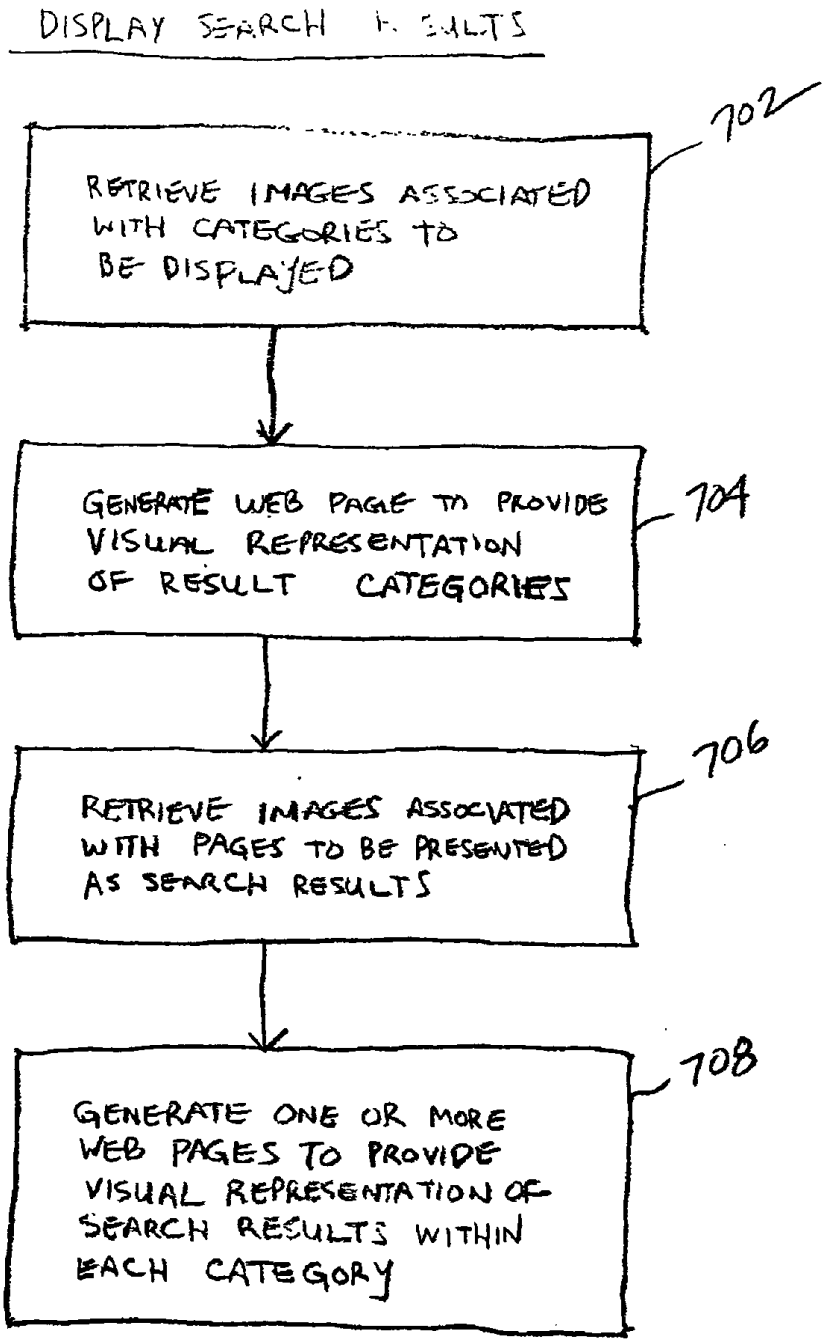

DISPLAY SEARCH RESULTS
RETRIEVE IMAGES ASSOCIATED WITH CATEGORIES TO BE DISPLAYED
702
GENERATE WEB PAGE TO PROVIDE VISUAL REPRESENTATION OF RESULT CATEGORIES
704
RETRIEVE IMAGES ASSOCIATED WITH PAGES TO BE PRESENTED AS SEARCH RESULTS
706
GENERATE ONE OR MORE WEB PAGES TO PROVIDE VISUAL REPRESENTATION OF SEARCH RESULTS WITHIN EACH CATEGORY
708

FIG. 7

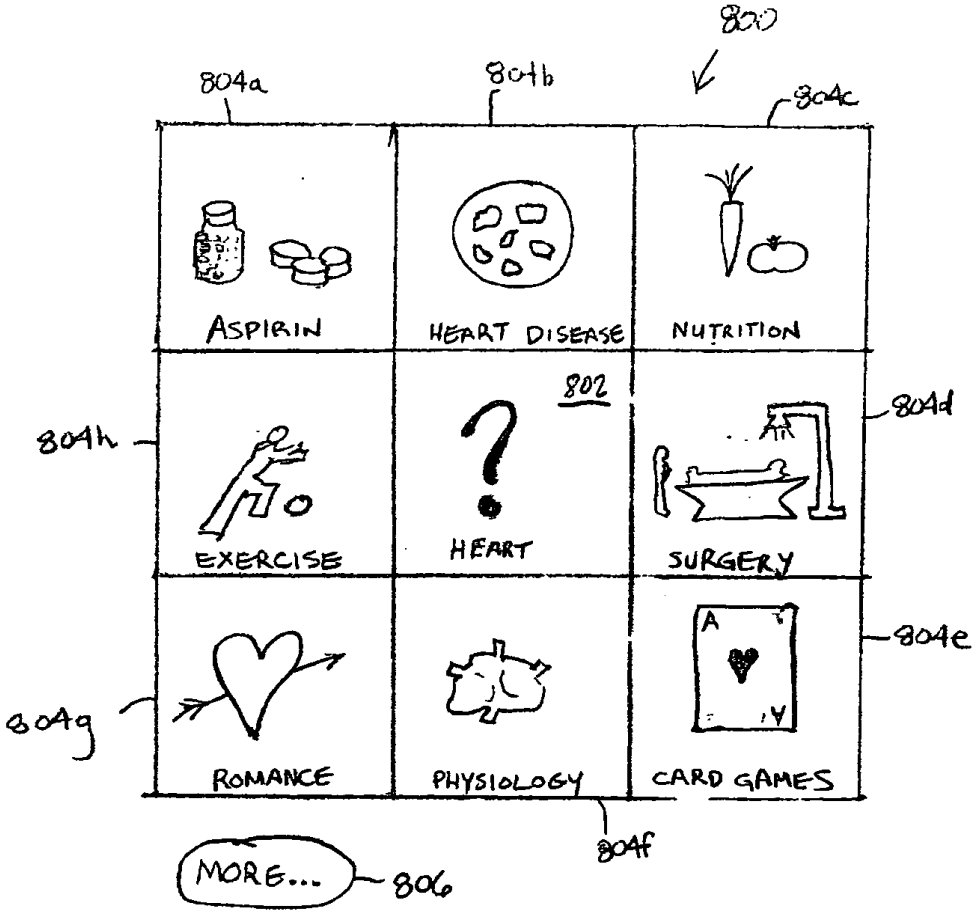

800
804a
804b
804c
ASPIRIN
HEART DISEASE
NUTRITION
802
804h
804d
EXERCISE
HEART
SURGERY
804e
804g
ROMANCE
PHYSIOLOGY
CARD GAMES
804f
MORE...
806

FIG. 8

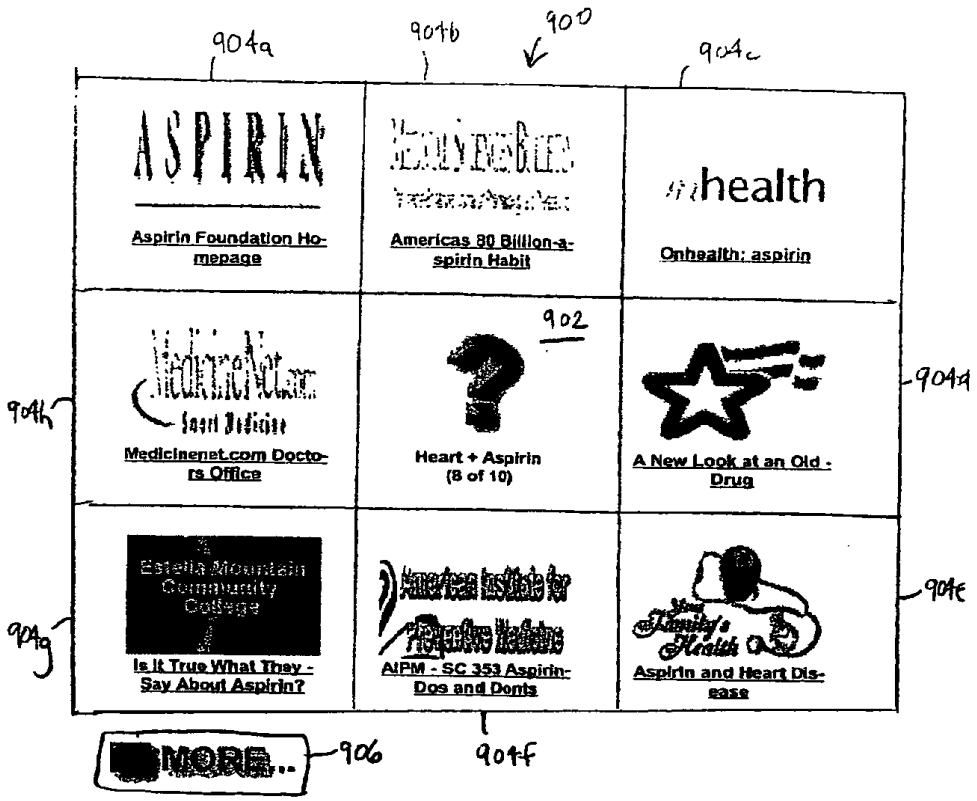

904a
904b
900
904c
ASPIRIN
Aspirin Foundation Homepage
Americas 80 Billion-a-spirin Habit
health
Onhealth: aspirin
902
MedicineNet.com Doctors Office
Heart + Aspirin (8 of 10)
A New Look at an Old - Drug
904d
904h
Estrella Mountain Community College
Is It True What They - Say About Aspirin?
AIPM - SC 353 Aspirin- Dos and Donts
Aspirin and Heart Disease
904e
904g
MORE...
906
904f

FIG. 9

INTERFACE FOR PRESENTING INFORMATION

FIELD OF THE INVENTION

[0001] The present invention relates generally to displaying search results. More specifically, providing a visual or multi-media representation of search results is disclosed.

BACKGROUND OF THE INVENTION

[0002] A variety of techniques for identifying records in a database that are responsive to a query submitted by a user are well known. One well known application of such techniques is their use in providing an Internet search engine to identify potentially relevant pages on the World Wide Web (referred to herein as "web pages") in response to a query submitted by a searching party.

[0003] It is well known that in order to be able to quickly identify web pages responsive to a query, one must first search tens of millions or hundreds of millions of the many millions of web pages accessible via the Internet and create a database containing information about each page. The information contained in such a database typically includes the address of the web page, such as the Uniform Resource Locator (URL) (i.e., the information a web browser would need to access the page) and one or more keywords associated with the page. The information in the database is used to identify web pages that may contain information that is responsive to a query submitted by a requesting party, such as by matching a term in a search query to a keyword associated with a web page.

[0004] A typical search engine presents results in the form of a list of responsive web pages. Each entry on the list typically corresponds to a web page, or a group of web pages from a single web site. Typically, a hypertext link is included for each web page listed. Text associated with the page also typically is provided, such as a brief description of the page, key words identified by the provider of the page, or excerpts of potentially relevant text that appears on the page.

[0005] In some cases, an effort is made to rank the results using a ranking scheme that is intended to result in the most relevant responsive pages being displayed on the list first. In some cases, well known statistical techniques are used to group at least certain of the responsive pages together into clusters or categories of responsive pages. In at least one case, such categories are displayed to the requesting party in the form of a folder icon for each category with an appropriate title or label on or near the folder icon. When a hypertext link associated with the folder icon is selected, the responsive web pages within the corresponding category are displayed in list form as described above.

[0006] The approaches described above for displaying search results have a number of shortcomings. First, the use of text to provide an indication to the requesting party of the content of web pages responsive to a query requires the requesting party to read the text associated with each page and determine whether the text indicates that the web page may contain the information the requesting party is seeking. This process may be time-consuming, depending on how long it takes the individual to read and comprehend the text provided for each responsive page and determine from the text whether or not the page contains the information sought, and how many such descriptions the individual must evaluate before the desired information is found or the individual either gives up or determines the search has not found any web page containing the desired information.

[0007] A second shortcoming of the above-described approach is that the text may not provide an accurate or complete indication of the true content of the web page. Much of the information available on the World Wide Web is provided in the form of images such as still pictures, video, audio, animated GIF's or other multimedia content. A textual description or excerpts of text from the page may not provide an adequate indication of such content and, at best, is an inefficient and time-consuming way to represent such content.

[0008] This second shortcoming has become even more apparent as increasing numbers of Internet users have gained access to broadband, high speed Internet connections, such as digital subscriber lines (DSL) and cable modem connections. The availability of such connections has accelerated the growth of multimedia content available on the Internet, increasing the need for an effective way to provide a representation of such content. Moreover, search engines that present search results in the form of a list of text entries do not take full advantage of the broadband connections now becoming available to an increasing number of users. Such connections make it possible to quickly and easily view search results displayed using a visual or multimedia representation of each site, such as a collage or slideshow of images, one or more video clips, and/or one or more audio clips from or associated with the content of the site.

[0009] Third, the approach described above can result in a tedious and potentially frustrating experience on the part of the requesting party. Reviewing a list of search results in the typical list form is much like reading a phone book or the entries in a card catalog. In many cases, a requesting party may review pages and pages of search results presented in such list form before the entry for the page having the desired information is found on the list. In some cases, the requesting party finds that the search has not identified a page having the desired information only after significant time has been spent reviewing search results in list form.

[0010] Finally, the approach described above results in a display that is static and not aesthetically pleasing. Many users are attracted to the Internet because of the visual, multi-media, and dynamic content available on the World Wide Web. Many users accustomed to such dynamic content find the typical search result list display described above to be both unfamiliar and uninteresting compared to other methods of displaying information on the World Wide Web.

[0011] It is critical to many providers of search engines that users find the site to be an interesting and aesthetically pleasing experience, as well as a useful and efficient way to find information. Search engine providers want to maximize the likelihood that a user will return to their site for further searches in the future. Advertising provides the only or most significant source of revenue for many such providers, and advertising revenue typically is based on the number of viewers, or "impressions", a site receives. As a result, search engine providers depend heavily for their commercial success on their ability to attract users to their site.

[0012] Search engines have been provided to locate images, video, music, and other multi-media content on the

Internet. The image, video, and/or music search engines provide by companies such as AltaVista™, Lycos™, and Ditto™ are typical. In some cases, the results of such searches have been presented in a form other than a list of web pages. In some cases, a thumbnail image of each responsive image retrieved from a database of images, such as images previously located on pages on the World Wide Web, is presented. However, in such cases the thumbnail image is used to represent the full-size image itself, not a web page the content of which is represented by the image, such as a web page that is responsive to a search query.

[0013] A visual interface has also been used to enable a user of the Internet to maintain a live HTML connection with more than one web site at a time by displaying multiple active web pages on a single display. Again, this technique has been used only to provide a split screen view for an Internet browser, and not to present a visual representation that quickly apprises a viewer of a display of the nature and content of a web page, such as a web page that is responsive to a search query.

[0014] It is also known to employ an advertising agency, graphical artist, or the like to create a set of images to be displayed in a slide show, such as in the banner advertisements that are ubiquitous on the Internet, to advertise a company, product, or service. In some cases, a link is provided in the banner ad to a web site associated with the company, product, or service. However, such slide shows have been used only to provide an advertising message or an inducement to attract users of the Internet to a web site associated with the company, product, or service being advertised. Such slide shows have not been used to our knowledge to provide a visual representation of the actual nature and content of a web page, such as a web page that is responsive to a search query.

[0015] Finally, it is known to provide for visual navigation through a site by enabling a user to select icons or images on one page in order to access additional or different information on another page. However, to our knowledge a visual interface has never been used to present the results of a search by providing a visual representation of web pages or categories of web pages, such as web pages or categories of web pages that are responsive to a search query.

[0016] Therefore, there is a need for a way to display search results in a manner that enables users to find records, such as web pages, having the information they are seeking quickly and efficiently. In addition, in the Internet environment there is a need for a way to display search results that makes use of the visual and multi-media content available on the World Wide Web. There is also a need to present search results in a way that is familiar and more satisfactory to users of the Internet. Finally, there is a need to present search results in a display that is dynamic, rather than static.

SUMMARY OF THE INVENTION

[0017] Accordingly, an interface for presenting search results is described. Responsive records are identified in response to a search query. Responsive records are grouped into categories of related responsive records, with a multimedia representation-such as a visual representation comprised of one or more images, animations, video segments, audio segments, or other multimedia content-being provided for each category. A multimedia representation of the nature and content of each responsive record within each category also is provided.

[0018] It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, a device, a method, or a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. Several inventive embodiments of the present invention are described below.

[0019] In one embodiment, a lexicon embodying information concerning words, phrases, and expression; their meaning; and their semantic and conceptual relations with each other is built. A database of images is collected. A database of pre-determined, or "static", search result categories is developed. One or more images is associated with each static category. Web pages on the World Wide Web are accessed. Each page is processed to identify a signature for the page and to harvest usable images from the page. Web page signatures and usable images are stored in a database. One or more images are associated with each web page. When a search query is received, Web pages responsive to the search query are identified. Responsive web pages are organized into categories of related responsive web pages. For each category and each responsive web page, one or more associated images are retrieved. The categories and responsive web pages are ranked. A display is provided to the requesting party in which one or more of the search result categories are represented by one or more associated images. By selecting a category, the requesting party accesses a display presenting one or more responsive web pages within the category.

[0020] Each responsive web page within a category is represented by one or more images associated with the web page. If one image is used, the display is static. If more than one image is used, the display is dynamic and the images alternate. In one embodiment, more than one image is used to represent each responsive web page and the images are arranged in a slideshow format.

[0021] In one embodiment, at least certain of the categories and/or certain of the responsive web pages are represented by one or more segments (or "clips") of video, audio, and/or other multimedia content. In one embodiment, at least certain of the responsive web pages are represented by one or more segments of video, audio, and/or other multimedia content harvested from the responsive web page.

[0022] In one embodiment, the disclosed interface is used in connection with a directory of information sources, such as the Open Directory Project on the Internet, to represent directory entries and categories of entries.

[0023] In one embodiment, a tag is used by the provider of a web page to identify the image(s), video, audio, or other multimedia content on the web page that the provider considers to be the most relevant for purposes of representing the nature and content of the web page. In one embodiment, a different tag is used for each type of multimedia content (e.g., one for each of static images, video, audio, etc.)

[0024] In one embodiment, a system and method are disclosed for presenting information. Categories are deter-

mined for found information by analyzing the content of the information. The categories are correlated with images that represent the categories. Images are displayed that correspond to the categories.

[0025] In one embodiment, a system and method are disclosed for presenting information. Textual content of the information is analyzed. The textual content is associated with image content. The image content is displayed to illustrate the information.

[0026] In one embodiment, a system and method are disclosed for building enriching content for a video presentation. Metadata related to the presentation is analyzed. Content is associated with the video presentation based on the analysis. The content is presented along with the video presentation.

[0027] These and other features and advantages of the present invention will be presented in more detail in the following detailed description and the accompanying figures which illustrate by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

[0029] **FIG. 1** is a block diagram illustrating a system used in one embodiment to provide a visual representation of search results.

[0030] **FIG. 2** is a flowchart of a process used in one embodiment to provide a visual representation of database search results in response to a user query.

[0031] **FIG. 3** is a block diagram illustrating the organization of a database **300** stored in database **106** of **FIG. 1** in one embodiment.

[0032] **FIG. 4** is a process flow showing in more detail a process used in one embodiment to implement step **204** of **FIG. 2**.

[0033] **FIG. 5** is a flowchart illustrating a process used in one embodiment to process web pages as described in step **206** of **FIG. 2**.

[0034] **FIG. 6** is a flowchart illustrating the process used in one embodiment to implement step **208** of **FIG. 2**.

[0035] **FIG. 7** is a flowchart illustrating a process used in a one embodiment to implement step **210** of **FIG. 2**.

[0036] **FIG. 8** is an exemplary search result categories display **800** used in one embodiment to display exemplary search result categories for a hypothetical search using the word "heart" as the search query.

[0037] **FIG. 9** is an exemplary responsive web pages display **900** used in one embodiment to implement step **708** of **FIG. 7**.

DETAILED DESCRIPTION

[0038] A detailed description of a preferred embodiment of the invention is provided below. While the invention is described in conjunction with that preferred embodiment, it should be understood that the invention is not limited to any one embodiment. On the contrary, the scope of the invention is limited only by the appended claims and the invention encompasses numerous alternatives, modifications and equivalents. For the purpose of example, numerous specific details are set forth in the following description in order to provide a thorough understanding of the present invention. The present invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the present invention is not unnecessarily obscured.

[0039] **FIG. 1** is a block diagram illustrating a system used in one embodiment to provide a visual representation of search results. One or more users **102** connect via the Internet with a search engine website system **100** used to provide a search engine web site by means of computer system **104** and database **106**. In one embodiment, computer system **104** comprises a super computer comprised of multiple computer processors and adequate memory, data storage capacity, and Internet bandwidth to provide search engine services via the Internet to multiple users simultaneously. In one embodiment, computer system **104** is configured to provide a web page via the Internet and to receive and process search queries received from users via the web page. The computer system **104** is connected to database **106** and is configured to store data in database **106** and to retrieve data stored in database **106**.

[0040] In one embodiment, computer system **104** is comprised of at least two computers. One computer is configured as a front end web server configured to provide a web page via the Internet capable of receiving search queries from users via the Internet. The front end web server performs the specialized task of presenting web pages to users and acting as an interface or conduit for information between the separate computer or computers used to process and generates results for search queries, on the one hand, and users of the web site, on the other hand. In such an embodiment, the logic functions necessary to process and provide results for search queries are performed by one or more additional computers configured as business logic servers. The front end web servers maintain a direct connection to the Internet and a connection to the business logic server or servers. The business logic server(s) in turn are connected to database **106** and are responsible for storing information to database **106** and retrieving information from database **106** to be processed by the business logic servers and/or to be provided to users via the front end web server(s).

[0041] The search engine website system **100** also is connected via the Internet to a plurality of web pages **110**, denominated as web page$_1$ through web page$_n$ in **FIG. 1**. Given the number of web pages currently available on the Internet, the number of web pages that may be accessible via a search engine such as one provided by search engine website system **100** may be on the order of tens of millions or hundreds of millions of web pages.

[0042] In order to be able to process search queries and identify responsive web pages, the computer system **104** is configured to access the web pages **110** in advance of receiving search queries from users **102** in order to build a database of information necessary to identify web pages that

are responsive to a search query and provide an efficient, useful, and visual representation of the search results. The computer system **104** may access the web pages **110** using any one of a number of readily available tools to perform that task, such as commercially available web crawler products that contain computer instructions necessary for a computer system such as computer system **104** to access a large number of web pages systematically by crawling from one page to the next, and so on. As each web page is accessed, information about the web pages is gathered and processed as described more fully below. The information gathered about the web pages **110** is stored in database **106** by computer system **104**.

[0043] **FIG. 2** is a flowchart of a process used in one embodiment to provide a visual representation of database search results in response to a user query. The process begins with a step **202** in which a lexicon and an image database are built. The lexicon **204** comprises a mapping of words, phrases and idiomatic expressions used in a given language, and their semantic, logical, and conceptual relationship to one another. In one embodiment, the lexicon **204** includes a mapping of collocations, i.e., the frequency with which words appear together in a language. Statistical natural language processing techniques for developing such a lexicon are well known in the art of linguistics. See, e.g., *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer,* by Gerard Salton (Addison Wesley Publisher Co., reprinted December 1988); *Foundations of Statistical Natural Language Processing,* by Christopher D. Manning and Hinrich Schüitze (MIT Press 1999); and *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon,* edited by Uri Zernik (Lawrence Erlbaum Assoc. 1991), which are hereby incorporated by reference for all purposes.

[0044] The lexicon is derived from a corpus of language content. The corpus is comprised of a very large body of content drawn from a wide variety of sources. The corpus may include raw content drawn from sources such as encyclopedias, newspapers, academic journals, and/or any of the multitude of content sources available on the Internet. Corpora developed for purposes of developing a lexicon through statistical natural language processing also are available. Some such corpora include tags or annotations that may be useful in building a lexicon, such as tags relating to sentence structure and tags identifying parts of speech. Automated statistical natural language processing techniques are applied to the corpus to build a lexicon to be used by search engine website system **100** of **FIG. 1**.

[0045] In one embodiment, the images database is comprised of images drawn from web pages accessed via the Internet. In one embodiment, the images database also includes images drawn from other sources, such as databases of images available on the Internet or commercially for use as clip art. In one embodiment, images generated by graphical designers or artists for the express purpose of being included in the images database also are included. In one embodiment, one or more images in the database are modified by adding a title, caption, or ticker associated with the image. Such metadata that is included with the image can be used to help determine a signature for each image. The image signature identifies the words, phrases, expressions, and concepts the image may be useful in representing. The image signature is stored. The image signature may be derived by noting the context of the page in which the image is displayed and assuming that the image is relevant to that context. The process continues with step **204** in which a database of static categories is created. As described more fully below, the static categories will be used, when suitable, to organize database records identified in response to a search query into categories for more convenient and efficient review by the user. The term "static" categories refers to the fact that the categories developed in step **204** are created in advance and do not change in response to a particular query or set of search results. As described more fully below, in one embodiment additional or different categories are created dynamically in some circumstances, such as where the responsive records cannot be grouped into a reasonable number of static categories that accurately described the content of the records in the group.

[0046] Next, in step **206**, individual web pages are processed to develop a signature for each page. The signature embodies information concerning the identity, location, nature, and content of each of the web pages **110** to be included in the database. In addition to developing a signature for each page, the images contained in each page are evaluated and, if usable, are added to the images database and associated with the web page as an image suitable for providing a visual representation of the content of the page. If no image, or an insufficient number of images, taken from a web page is identified as suitable for providing a visual representation of the content of the page, other images from the images database, or a picture of the web page itself as viewed by a browser, are associated with the page.

[0047] Then, in step **208**, a search query is received from the user and processed. Responsive web pages are identified and grouped into appropriate static and/or dynamic categories and result categories and responsive web pages within each category are ranked, as described below.

[0048] Finally, in step **210**, search results are displayed to the user using a visual representation described more fully below.

[0049] **FIG. 3** is a block diagram illustrating the organization of a database **300** stored in database **106** of **FIG. 1** in one embodiment. The database includes a corpus database **302** used to store the corpus described above. The database also includes a lexicon **304**, built using the corpus, as described above. The third component of the database **300** is the images database **306**.

[0050] The database **300** also includes a categories database **308**. Finally, the database **300** includes a web page signatures database **310** in which the signature of each web page and an identification of the image(s) associated with the web page are stored.

[0051] **FIG. 4** is a process flow showing in more detail a process used in one embodiment to implement step **204** of **FIG. 2**. The process begins with a step **402** in which a database of static search categories and associated subcategories is built. An effort is made to anticipate the topics, types of search, and types of information users may be interested in finding by means of queries submitted to the search engine website. The lexicon described above is used in one embodiment to develop categories and associated subcategories that may be useful in presenting search results. In one embodiment, the lexicon is used to identify

words, phrases, and/or expressions that have a close semantic or conceptual relationship with a word or combination of words that it is anticipated may be included in a query. These related words, phrases, and/or expressions are then stored as static categories and subcategories associated with the word or combination of words.

[0052] Next, in step **404**, at least one image from the image database is associated with each category or subcategory stored in the category database. As noted above, when images are stored in the image database, information about the image and the words and concepts the image may be appropriate to represent also are stored in the database. This information is used to match images from the database with corresponding categories and subcategories in the category database so that an image may be used to provide a visual representation of the category to a user.

[0053] **FIG. 5** is a flowchart illustrating a process used in one embodiment to process web pages as described in step **206** of **FIG. 2**. Each step in the flowchart shown in **FIG. 5** is performed with respect to each web page accessed in the manner described above, such as using a web crawler. The process begins with step **502** in which the web page is accessed. Next, in step **504**, the page is analyzed to generate a signature for the page. This process includes the application of well known statistical natural language processing techniques to the text content of the web page to identify the words, subjects, and concepts that are the primary, or a significant, focus of the content of the page.

[0054] In addition, the HTML (hypertext markup language) or other computer code used to display the web page to those accessing the web page is analyzed to extract information about the page that may not be available from the text content of the page itself. For example, computer programming languages such as HTML provide a way to tag information in the code, such as to indicate the meaning, nature, or significance of the information. A standard setting body establishes standards for the use of such tags to annotate the code. One well known application of such tags is the use of a tag to identify keywords that the providers of the page believe describe the nature and content of the page. Such keywords may be used, in addition to information derived from the natural language processing techniques referred to above, to develop a signature for the page. The signature will later be used to identify pages responsive to a query from a user.

[0055] The process continues with step **506** in which the images included in the web page are identified and evaluated. In one embodiment, all GIF and JPEG files on a web page, and all code associated with such files, is evaluated. GIF and JPEG files are commonly used to provide graphical images on web pages. In one embodiment, an automatic parsing algorithm is used to determine whether each image on a web page may be suitable to be added to the images database, either for use in representing a category or subcategory of information, or to be used to provide a visual representation of the content of either the page from which it is harvested or another web page that contains information related to the image but that does not itself have images suitable for use in representing the page. The properties of each image that are evaluated include the location of the image within the page, whether the image has a subject or title associated with it, the way the image is referred to in the text on the web page, and the size of the image and its associated computer file. For example, an image that is relatively large, centrally located, and annotated with a title or caption that correlates with the signature of the page may be selected as an image suitable for representing the content of the page. By contrast, an image that is small, has no text associated with it, and appears on the bottom or periphery of the web page may be rejected.

[0056] In step **508**, images on the page that may be usable to represent either a search category, the page itself, or some other page are harvested from the page and stored in the images database. As noted above, a signature for the image also is stored.

[0057] Next, in step **510** the overall appearance of the web page itself is evaluated to determine whether a picture of the entire web page should be captured and stored in the database. For example, a web page that contains a large image or several images closely related to the signature for the web page may be represented visually by a reduced size image of the entire web page. Products and services for obtaining such reduced size images of entire web pages are available commercially, including products and services that provide a GIF capture of a target web page.

[0058] In one embodiment, the above-described techniques for identifying images in a web page that may be suitable for providing a visual representation of the web page are replaced or augmented by enabling providers of web pages to identify the images on the page that the provider believes are the most relevant or useful. For example, providers could be provided with a way to tag the HTML or other code used to provide the page in a manner that identifies the image or images on the web page that the provider of the web page believes are the most relevant or important images on the page, or the ones most suitable to be used to provide a visual representation of the page such as to present search results. A standard for such tagging of images has not yet been provided, but could readily be established by the standard setting bodies for languages, such as HTML, that are commonly used to provide web pages. For example, such a standard could easily be modeled on the standard that currently enables providers of web pages to identify keywords for a web page.

[0059] The process shown in **FIG. 5** concludes with step **512** in which one or more images form the images database are associated with the web page. Preferably, the images associated with the web page will be images harvested from the page itself. However, in cases where the web page itself did not have a sufficient number of images suitable for use in providing a visual representation of the page as a search result, as described above other images from the images database having a signature or description that matches the signature of the page may be drawn from the images database to be associated with the web page for future use in providing a visual representation of the page.

[0060] In one embodiment, a score is assigned to the web page and stored in the web page signature database to provide an indication of the extent to which the page contains high quality images and/or other media content that is relevant to the main information contained in the page. In one embodiment, this assessment of the visual and/or multimedia content of each web page is used, among other factors, to determine a relative ranking for each web page

identified as responsive to a query. Using this approach, web pages that are rich in visual and/or multi-media content are more likely to receive a higher ranking and, therefore, to appear in one of the first several layers or pages of search results presented to the requesting party. In many cases, this approach will result in a search results display that is more visually interesting and familiar to the requesting party.

[0061] **FIG. 6** is a flowchart illustrating the process used in one embodiment to implement step **208** of **FIG. 2**. The process begins with step **602** in which a search query is received from a user. Next, in step **604** the query is analyzed to determine the words, phrases, expressions, and concepts most closely associated with the word or combination of words provided by the user in the query. Next, in step **606** the database of web page signatures is searched to identify web pages having a signature that matches in whole or in part the word or combination of words in the query.

[0062] Then, in step **607**, tentative search result categories are generated dynamically using collocations. That is, the lexicon is used to identify words or phrases that often appear together with one or more search terms or phrases. Next, in step **608**, it is determined whether the categories generated based on the collocations are satisfactory. The signatures of the responsive web pages are searched to determine if the collocations are associated with a significant portion of the web pages such that the collocations provide a satisfactory means of grouping the results (e.g., by defining a manageable number of categories that include most of the web pages and with sufficient distribution of pages among the categories).

[0063] If the categories based on collocations are satisfactory, the process proceeds to step **614**, in which the categories are ranked in terms of how closely they are related to the query. Also, the responsive web pages within each category are ranked within the category based on how closely the signature for each web page matches the query. Specific techniques for performing such ranking are well known in the art and are beyond the scope of this disclosure.

[0064] If the categories based on collocations are not satisfactory, the process continues with step **609**, in which an attempt is made to associate the responsive web pages with previously-defined categories from the categories database. In one embodiment, the categories most closely related to the signature for each web page are identified and assigned a weight indicating how closely the category matches the signature. The weighted static categories are then evaluated in step **610** to determine if the responsive web pages can be grouped within a reasonable number of static categories that will both encompass a sufficient number of the web pages and describe the nature and content of the web pages within each group adequately. In one embodiment, the weighted static categories are evaluated to determine whether the responsive results may be represented adequately by from one to ten static categories.

[0065] If the static categories do provide a satisfactory grouping and representation of the responsive web pages, the process proceeds to step **614** in which the categories and responsive web pages are ranked. If in step **610** it is determined that the matching of responsive web pages to static categories has not resulted in a satisfactory grouping and representation of the search results, the process proceeds to step **612** in which well known statistical techniques are used to group the responsive web pages into clusters of related responsive web pages based on the signature of each page. Statistical natural language processing techniques are then used to generate a category name dynamically for each cluster. Then, the process proceeds to step **614**, in which the dynamically generated categories are ranked and the web pages within each category are ranked, as described above.

[0066] The process begins with step **702** in which images associated with the categories to be displayed are retrieved from the images database. Next, in step **704**, a web page is generated to provide a visual representation of the result categories. Then, in step **706**, the images associated with the web pages to be presented as search results are retrieved from the images database. Finally, in step **708**, one or more web pages are generated to provide a visual representation of the responsive web pages within each category.

[0067] **FIG. 8** is an exemplary search result categories display **800** used in one embodiment to display exemplary search result categories for a hypothetical search using the word "heart" as the search query. As shown in **FIG. 8**, the search result categories display **800** is divided into a 3×3 grid of 9 cells. The center cell **802** contains an image of a question mark and the text of the search query, in this case the word "heart". The remaining 8 cells of the grid, cells **804*a*-804*h***, are used to provide a visual representation of the eight top ranked search result categories. The exemplary categories shown in **FIG. 8** include the categories "aspirin", "heart disease", "nutrition", "surgery", "card games", "physiology", "romance", and "exercise". In each of cells **804*a*-804*h***, the name of the category displayed in the cell is listed at the bottom of the cell and an image that provides a visual representation of the result category is displayed in the cell above the category name. The search result categories display **800** also includes a button **806** which, when selected, will result in the next eight categories by rank (or the remaining categories, if less than eight remain) being displayed in the search results categories display **800**. While the exemplary categories display **800** presents eight categories at a time, it is readily apparent that any number of categories may be displayed at one time, and that geometries other than the 3×3 grid geometry show in **FIG. 8**, such as a hub and spoke arrangement, can be used.

[0068] The search results categories display **800** provides an efficient and aesthetically pleasing way for the user to find and access the responsive web pages that are most likely to contain the information the requesting party is seeking. For example, a requesting party interested in the latest information available about the benefits and risks of taking aspirin as a preventive measure prior to the onset of heart disease would be drawn quickly to the image of a bottle of aspirins and several aspirin tablets displayed in cell **804*a*** of **FIG. 8**. The requesting party likewise would be able to quickly filter out wholly irrelevant information, such as web pages grouped under the category "romance", by recognizing that the image of the heart shape with an arrow through it is an image related to the heart as a symbol of romantic love, and not a health-related concept.

[0069] **FIG. 9** is an exemplary responsive web pages display **900** used in one embodiment to implement step **708** of **FIG. 7**. The responsive web pages display **900** shown in **FIG. 9** is a continuation of the example described above with respect to **FIG. 8** in which the user has selected the

category "aspirin". The responsive web pages display **900** is divided into a 3×3 grid of 9 cells, similar to the display **800** in **FIG. 8**. The center cell **902** contains the same question mark image as center cell **802** in **FIG. 8**. The text that appears beneath the image in center cell **902** indicates that the responsive web pages display **900** is being used to display web pages responsive to a query comprised of the search term "heart" that have been grouped within the category named "aspirin". The text also indicates that the display is being used to show eight of ten responsive websites in the category being displayed.

[0070] In the outer cells **904*a*-904*h***, each cell is used to provide a visual representation of one of the eight top ranked responsive web pages within the category "aspirin". In one embodiment, a single representative image previously associated with each web page appears in the cell corresponding to the responsive web page. In one embodiment, multiple images are associated with each web page in the database and an animated slide show of images associated with the web page is presented for each web page displayed. As shown in **FIG. 9**, in one embodiment, text appears beneath the image or images displayed for each web page describing the nature, location, source, and/or content of the responsive web page.

[0071] The responsive web pages display **900** also includes a more pages button **906** which, when selected, results in the next zero to eight responsive web pages being displayed. In the case illustrated in **FIG. 9**, only two additional websites within the category "aspirin" would be displayed.

[0072] In one embodiment, the slide show images are rotated at relatively slow intervals when the cursor is not on a particular one of cells **904*a*-904*h*** and the pace of the slide show accelerates appreciably when the cursor is placed on a particular one of cells **904*a*-904*h***. This permits the requesting party to quickly view the set of images associated with a particular responsive web page by placing the cursor on the slide show for that page.

[0073] The above-described visual representation of search result categories and responsive web pages enables users to find desired information more quickly and efficiently by using a visual interface, which is much more familiar to users of the Internet than the traditional list approach. In addition, the slide show approach is advantageous because it enables a requesting party to do the equivalent of flipping through pages of a book or magazine on a bookshelf in a bookstore. By viewing the slide show, a requesting party can quickly get a sense of the nature of a web page and the content the user will find if the user accesses the page. By contrast, when search results are presented in a list or folder format, a requesting party must spend time reading a written description of each web page that may or may not provide an accurate indication of the content of the web page. Furthermore, the above-described approach saves on the number of mouse or other pointer "clicks" needed to review search results and find information, as a user can in many cases get more complete information regarding the multimedia content of a page without actually visiting the page.

[0074] It should be noted that while the above detailed description focuses on a particular embodiment in which images are used to provide a visual representation of search result categories and responsive web pages, it is contemplated that the approach described above will be used with other forms of content available in sources of information such as the Internet. For example, there is a wealth of video content available on the Internet. Such video content could be accessed, evaluated, and harvested in the same manner as described above for static images. Harvested video could be associated with search result categories and web pages as described above with respect to the static images, and used in displays similar to those shown in **FIGS. 8 and 9** to represent search categories and responsive web pages respectively.

[0075] In such a video embodiment, segments of video would be selected to represent search result categories or responsive web pages in the same manner as described above for static images. The video clips would then be presented in reduced form in the same manner as shown in **FIGS. 8 and 9**. Such video clips would have the same advantage as static images, presented either singly or in a slide show as described above, in permitting a requesting party to quickly determine which categories of information and which responsive web pages within categories of interest are most likely to contain the information the requesting party is seeking. Audio clips likewise can be used to provide a multimedia representation of the nature and content of a web page in the same manner as described above with respect to images and video.

[0076] While the above description focuses on an embodiment in which the database being searched is a database of web pages available via the Internet, the approach is equally applicable to presenting search results in response to a query of any database of information in which the database records may be represented by an associated image or set of images. Contemplated applications include interactive television applications. For example, a viewer of a sporting event on television may be provided with a cursor or other pointing device to be used to select images on the screen concerning which the requesting party would like to retrieve additional information. Alternatively, a viewer may be provided with a means for entering a search query in the form of text related to a program the viewer is viewing. In either case, a visual representation of search results such as those shown in **FIGS. 8 and 9**, and described above would be an advantageous and visually pleasing way to present search results on the television screen to such a viewer.

[0077] In another interactive television embodiment, a database of information is accessed to provide a parallel presentation to a television broadcast or video presentation. Information about the broadcast is derived by either analyzing the broadcast or metadata associated with the broadcast such as a datacast and querying the database based on what is being broadcast to find and present information that is related to the broadcast. For example, close caption information associated with the broadcast may be used to determine the broadcast content and search for related material.

[0078] In other embodiments, the search techniques described above may be used to search for and present material included on a DVD or other medium in addition to material found on the Internet.

[0079] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modi-

fications may be practiced. It should be noted that there are many alternative ways of implementing both the process and apparatus of the present invention. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein.

What is claimed is:

1. A method of presenting a search result comprising:

determining categories for found information by analyzing the content of the information;

correlating the categories with images that represent the categories; and

displaying images that correspond to the categories.

2. A method of presenting a search result as recited in claim 1 wherein images corresponding to the found information are displayed when a user activates one of the categories.

3. A method of presenting a search result as recited in claim 2 wherein the user activates one of the categories by dragging a cursor over the image that corresponds to the category.

4. A method of presenting a search result as recited in claim 1 wherein the display is a grid.

5. A method of presenting a search result as recited in claim 1 wherein the information includes a plurality of web sites.

6. A method of presenting a search result as recited in claim 5 further including providing a rotating display of content from the web sites.

7. A method of presenting a search result as recited in claim 5 further including providing a video display of content from the web sites.

8. A method of presenting a search result as recited in claim 5 further including rating each web site according to whether the web site includes image content that is relevant to textual content on the web site.

9. A method of presenting a search result as recited in claim 1 wherein the information includes information stored on a DVD.

10. A method of presenting a search result as recited in claim 6 wherein dynamically displaying content from the web sites includes showing representative images from the web site that correspond to textual content in the web site.

11. A system for presenting a search result comprising:

a processor configured to determine categories for found information by analyzing the content of the information;

a database containing images that correspond to the categories; and

a processor configured to generate a display of images that correspond to the categories.

12. A computer program product for presenting a search result, the computer program product being embodied in a computer readable medium and comprising computer instructions for:

determining categories for found information by analyzing the content of the information;

correlating the categories with images that represent the categories; and

displaying images that correspond to the categories.

13. A method of presenting information comprising:

analyzing textual content of the information;

associating the textual content with image content; and

displaying the image content to illustrate the information.

14. A method of presenting information as recited in claim 13 wherein the image content is included in the information.

15. A method of presenting information as recited in claim 13 wherein the image content is not included in the information.

16. A method of presenting information as recited in claim 13 wherein metadata associated with the image content is correlated with the textual content to determine the image content that is associated with the textual content.

17. A method of presenting information as recited in claim 13 wherein the information includes a web site.

18. A method of summarizing a web site comprising:

reading tags associated with a web site wherein certain of the tags indicate that material associated with the tags is representative material; and

displaying the representative material as a representative of the website.

19. A method of summarizing a web site as recited in claim 18 further including displaying the representative material in response to a search request.

20. A computer program product for presenting information, the computer program product being embodied in a computer readable medium and comprising computer instructions for:

analyzing textual content of the information;

associating the textual content with image content; and

displaying the image content to illustrate the information.

21. A system for presenting information comprising:

a processor configured to analyze textual content of the information and associate the textual content with image content; and

a display configured to display the image content to illustrate the information.

22. A method of building enriching content for a video presentation comprising:

analyzing metadata related to the presentation;

associating content with the video presentation based on the analysis; and

presenting the content along with the video presentation .

23. A method of building enriching content for a video presentation as recited in claim 22 wherein the metadata is close caption information.

24. A method of building enriching content for a video presentation as recited in claim 22 wherein the metadata is obtained from datacasting.

25. A method of building enriching content for a video presentation as recited in claim 22 wherein the content is downloaded from the Internet.

26. A method of building enriching content for a video presentation as recited in claim 22 wherein the video presentation is presented in an interactive television system.

27. A computer program product for building enriching content for a video presentation, the computer program

product being embodied in a computer readable medium and comprising computer instructions for:

analyzing metadata related to the presentation;

associating content with the video presentation based on the analysis; and

presenting the content along with the video presentation .

28. A system for building enriching content for a video presentation comprising:

a processor configured to analyze metadata related to the presentation and associate content with the video presentation based on the analysis; and

a display configured to present the content along with the video presentation.

* * * * *

US006553385B2

(12) **United States Patent**
Johnson et al.

(10) **Patent No.: US 6,553,385 B2**
(45) **Date of Patent: *Apr. 22, 2003**

(54) **ARCHITECTURE OF A FRAMEWORK FOR INFORMATION EXTRACTION FROM NATURAL LANGUAGE DOCUMENTS**

(75) Inventors: **David E. Johnson**, Cortlandt Manor, NY (US); **Thomas Hampp-Bahnmueller**, Tuebingen (DE)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/145,408**

(22) Filed: **Sep. 1, 1998**

(65) **Prior Publication Data**

US 2002/0007358 A1 Jan. 17, 2002

(51) **Int. Cl.**[7] **G06F 17/00**
(52) **U.S. Cl.** **707/104.1**; 704/9
(58) **Field of Search** 707/1, 2, 3–6, 707/104.1; 706/45; 704/9; 709/328

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,736,320	A *	4/1988	Bristol	395/703
4,965,763	A *	10/1990	Zamora	704/1
5,371,807	A *	12/1994	Register et al.	382/159
5,680,628	A *	10/1997	Carus et al.	704/1
5,682,539	A *	10/1997	Conrad et al.	704/9
5,974,412	A *	10/1999	Hazlehurst et al.	707/3
5,991,710	A *	11/1999	Papineni et al.	704/2
6,006,221	A *	12/1999	Liddy et al.	707/5
6,052,693	A *	4/2000	Smith et al.	707/104
6,070,133	A *	5/2000	Brewster et al.	704/9
6,076,088	A *	6/2000	Paik et al.	707/5
6,081,773	A *	6/2000	Hirai et al.	704/3

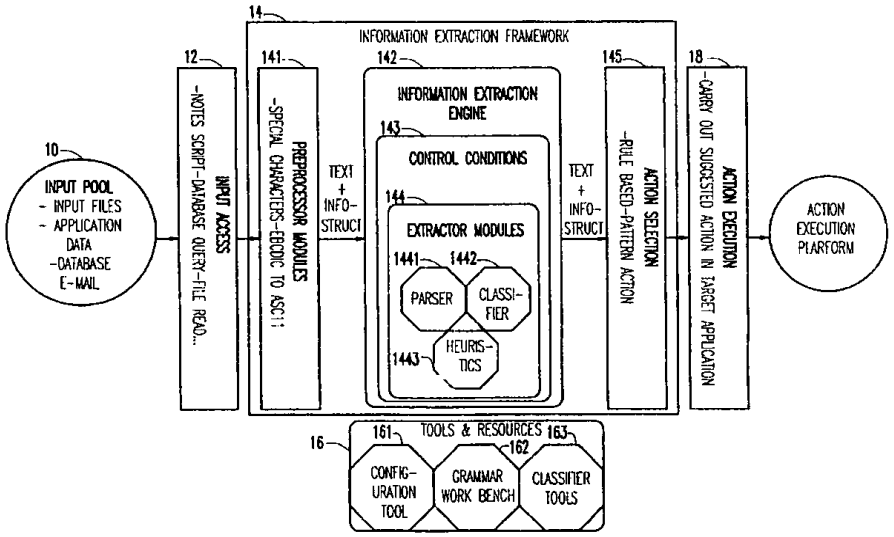
* cited by examiner

Primary Examiner—Safet Metjahic
Assistant Examiner—Uyen Le
(74) *Attorney, Agent, or Firm*—Whitham, Curtis & Christofferson, P.C.; Stephan C. Kaufman

(57) **ABSTRACT**

A framework for information extraction from natural language documents is application independent and provides a high degree of reusability. The framework integrates different Natural Language/Machine Learning techniques, such as parsing and classification. The architecture of the framework is integrated in an easy to use access layer. The framework performs general information extraction, classification/categorization of natural language documents, automated electronic data transmission (e.g., E-mail and facsimile) processing and routing, and plain parsing. Inside the framework, requests for information extraction are passed to the actual extractors. The framework can handle both pre- and post processing of the application data, control of the extractors, enrich the information extracted by the extractors. The framework can also suggest necessary actions the application should take on the data. To achieve the goal of easy integration and extension, the framework provides an integration (outside) application program interface (API) and an extractor (inside) API.

13 Claims, 3 Drawing Sheets



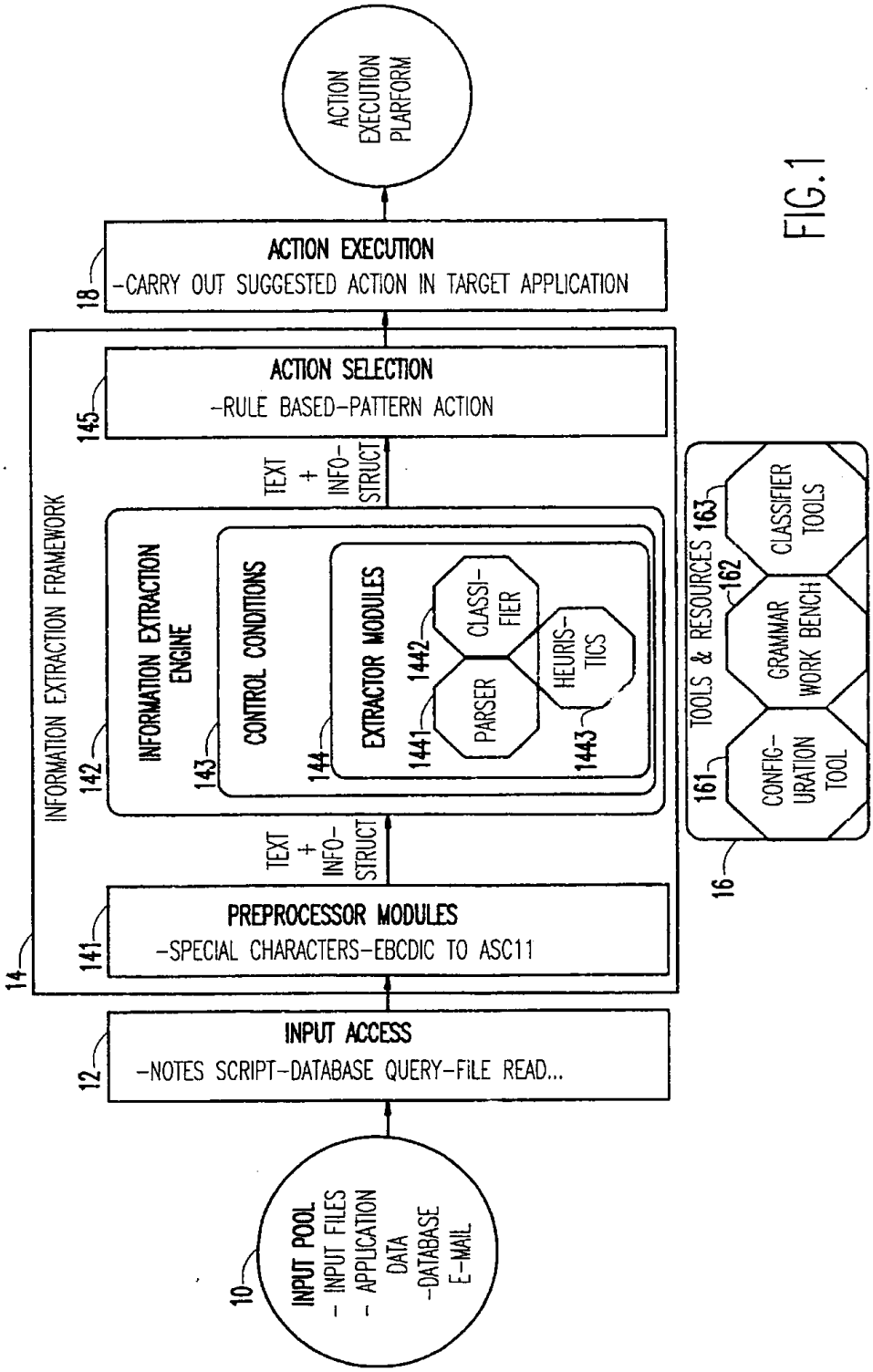

ACTION EXECUTION PLARFORM
ACTION EXECUTION
-CARRY OUT SUGGESTED ACTION IN TARGET APPLICATION
18
ACTION SELECTION
-RULE BASED-PATTERN ACTION
145
TEXT + INFO-STRUCT
INFORMATION EXTRACTION FRAMEWORK
INFORMATION EXTRACTION ENGINE
142
CONTROL CONDITIONS
143
EXTRACTOR MODULES
144
PARSER
1441
CLASSI-FIER
1442
HEURIS-TICS
1443
TOOLS & RESOURCES 163
CLASSIFIER TOOLS
GRAMMAR WORK BENCH
162
CONFIG-URATION TOOL
161
16
TEXT + INFO-STRUCT
PREPROCESSOR MODULES
-SPECIAL CHARACTERS-EBCDIC TO ASC11
141
14
INPUT ACCESS
-NOTES SCRIPT-DATABASE QUERY-FILE READ...
12
INPUT POOL
- INPUT FILES
- APPLICATION DATA
-DATABASE
E-MAIL
10

FIG. 1

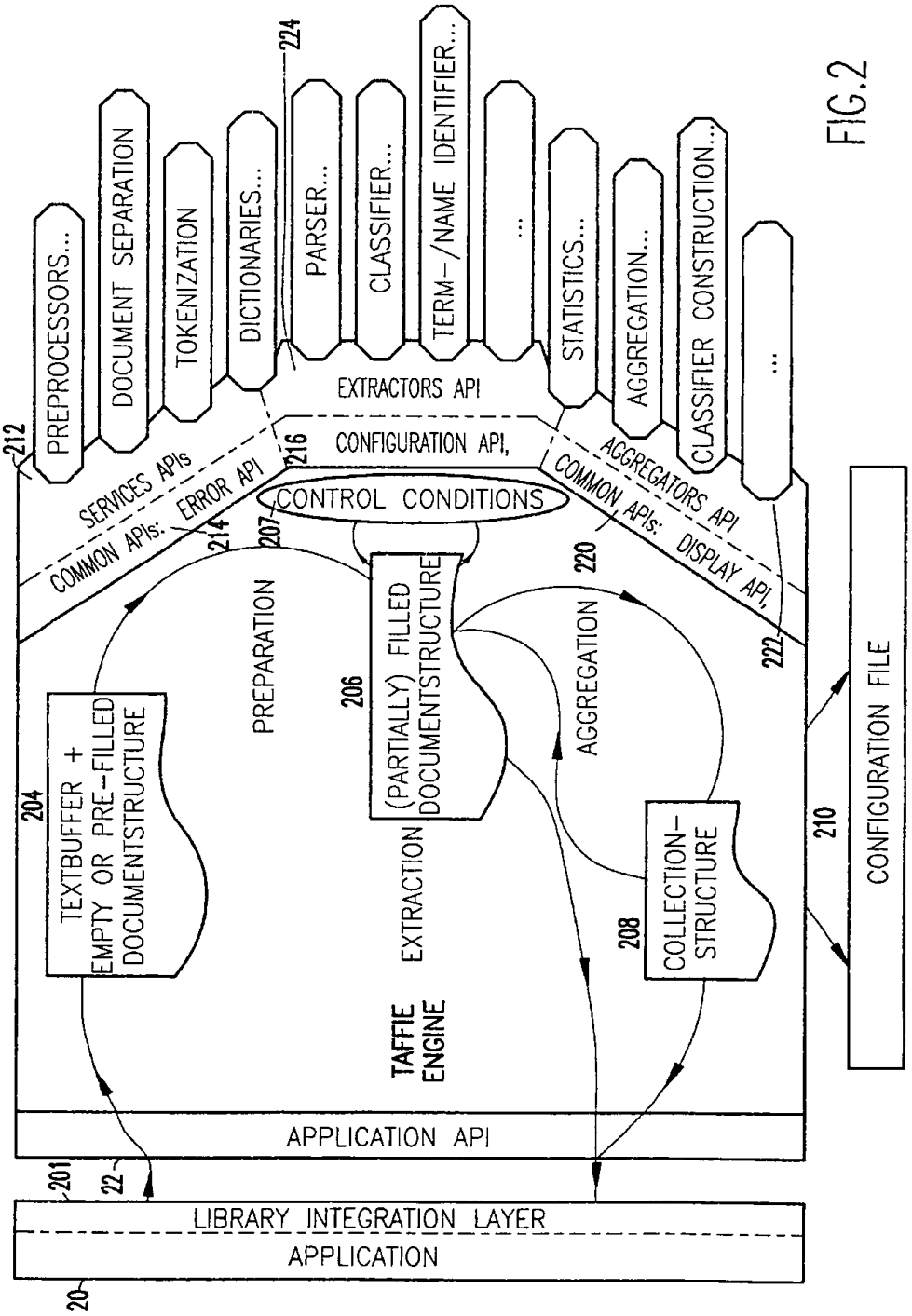

PREPROCESSORS...
DOCUMENT SEPARATION
TOKENIZATION
DICTIONARIES...
PARSER...
CLASSIFIER...
TERM-/NAME IDENTIFIER...
...
STATISTICS...
AGGREGATION...
CLASSIFIER CONSTRUCTION...
...
224
212
EXTRACTORS API
SERVICES APIs
COMMON APIs: ERROR API
216
CONFIGURATION API,
CONTROL CONDITIONS
214
207
AGGREGATORS API
COMMON APIs: DISPLAY API,
220
222
PREPARATION
206
(PARTIALLY) FILLED DOCUMENTSTRUCTURE
AGGREGATION
204
TEXTBUFFER + EMPTY OR PRE-FILLED DOCUMENTSTRUCTURE
EXTRACTION
208
COLLECTION-STRUCTURE
210
CONFIGURATION FILE
TAFFIE ENGINE
APPLICATION API
201
22
LIBRARY INTEGRATION LAYER
APPLICATION
20

FIG.2

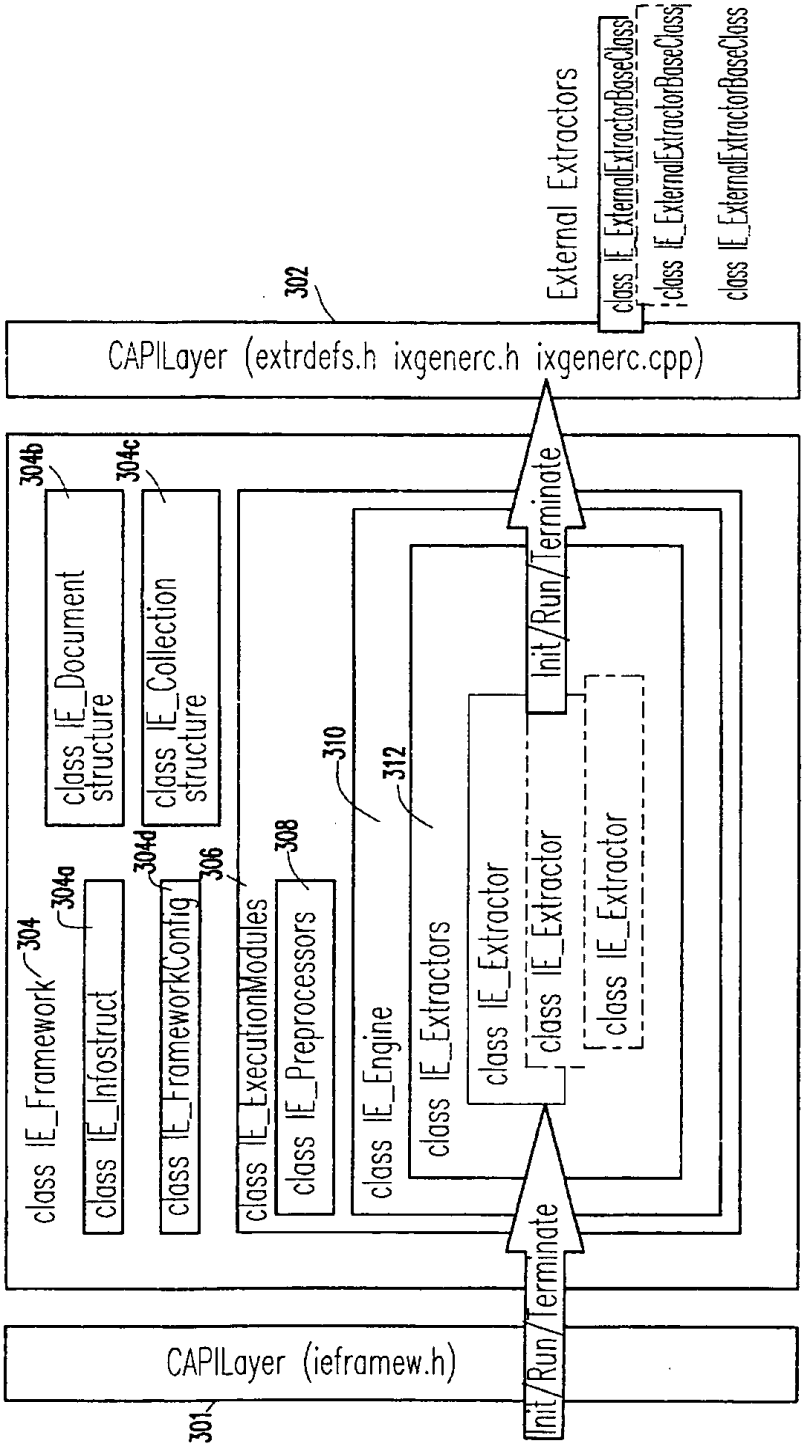

External Extractors
class IE_ExternalExtractorBaseClass
class IE_ExternalExtractorBaseClass
class IE_ExternalExtractorBaseClass
302
CAPILayer (extrdefs.h ixgenerc.h ixgenerc.cpp)
Init/Run/Terminate
304b
class IE_Document structure
304c
class IE_Collection structure
310
312
304
class IE_Framework
304a
class IE_Infostruct
304d
class IE_FrameworkConfig
306
class IE_ExecutionModules
308
class IE_Preprocessors
class IE_Engine
class IE_Extractors
class IE_Extractor
class IE_Extractor
class IE_Extractor
Init/Run/Terminate
CAPILayer (ieframew.h)
301

FIG.3

ARCHITECTURE OF A FRAMEWORK FOR INFORMATION EXTRACTION FROM NATURAL LANGUAGE DOCUMENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to knowledge information processing and, more particularly, to a general architecture of a framework for information extraction from natural language (NL) documents. The framework can be configured and integrated in applications and may be extended by user built information extractors.

2. Background Description

Businesses and institutions generate many documents in the course of their commerce and activities. These are typically written for exchange between persons without any plan for machine storage and retrieval. The documents, for purposes of differentiation, are described as "natural language" documents as distinguished from documents or files written for machine storage and retrieval.

Natural language documents have for some time been archived on various media, originally as images and more recently as converted data. More specifically, documents available only in hard copy form are scanned and the scanned images processed by optical character recognition software to generate machine language files. The generated machine language files can then be compactly stored on magnetic or optical media. Documents originally generated by a computer, such as with word processor, spread sheet or database software, can of course be stored directly to magnetic or optical media. In the latter case, the formatting information is part of the data stored, whereas in the case of scanned documents, such information is typically lost.

There is a significant advantage from a storage and archival stand point to storing natural language documents in this way, but there remains a problem of retrieving information from the stored documents. In the past, this has been accomplished by separately preparing an index to access the documents. Of course, the effectiveness of this technique depends largely on the design of the index. A number of full text search software products have been developed which will respond to structured queries to search a document database. These, however, are effective only for relatively small databases and are often application dependent; that is, capable of searching only those databases created by specific software applications.

The natural language documents of a business or institution represents a substantial resource for that business or institution. However, that resource is only [a] as valuable as the ability to access the information it contains. Considerable effort is now being made to develop software for the extraction of information from natural language documents. Such software is generally in the field of knowledge based or expert systems and uses such techniques as parsing and classifying. The general applications, in addition to information extraction, include classification and categorization of natural language documents and automated electronic data transmission processing and routing, including E-mail and facsimile.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a framework for information extraction from natural language documents which is application independent and provides a high degree of reusability.

It is another object of the invention to provide a framework for information extraction which integrates different Natural Language/Machine Learning techniques, such as parsing and classification.

According to the invention, there is provided an architecture of a framework for information extraction from natural language documents which is integrated in an easy to use access layer. The framework performs general information extraction, classification/categorization of natural language documents, automated electronic data transmission (e.g., E-mail and facsimile) processing and routing, and parsing.

Inside the framework, requests for information extraction are passed to the actual extractors. The framework can handle both pre- and post processing of the application data, control of the extractors, enrich the information extracted by the extractors. The framework can also suggest necessary actions the application should take on the data. To achieve the goal of easy integration and extension, the framework provides an integration (outside) application program interface (API) and an extractor (inside) API. The outside API is for the application program that wants to use the framework, allowing the framework to be integrated by calling simple functions. The extractor API is the API for doing the actual processing. The architecture of the framework allows the framework to be extended by providing new libraries exporting certain simple functions.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

FIG. **1** is a block diagram of the general layout of the architecture of a framework for information extraction from natural language documents according to the present invention;

FIG. **2** is a block diagram of the text analysis framework for information extraction according to the invention; and

FIG. **3** is a block diagram showing the object inclusion and flow of control in the information extraction framework.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

Referring now to the drawings, and more particularly to FIG. **1**, the general layout of the architecture of the framework for information extraction according to the invention will first be described. The framework provides tools for information extraction and natural language processing integrated in an easy to use access layer. This enables adding an information extraction component to an application by interfacing to the framework via an API.

Inside the framework, requests for information extraction are passed on to the actual extractors. The framework can handle pre- and postprocessing of the application data, control of the extractors and it also enriches the information extracted by the extractors. It also can suggest necessary actions the application should take on the data.

To achieve the goal of easy integration and extension, the framework provides two interfaces:

1. Integration (outside) API: This is the API for the application that wants to use the framework. The framework can be integrated by calling some simple functions that are provided by the framework library.
2. Extractor (inside) API: This is the API for the extractors doing the actual processing. The framework can be

extended by providing new libraries exporting certain simple functions.

The input pool **10** to the framework **14** may comprise input files, application data, data from a database, E-mail, facsimile, or the like. The output from this input pool is "raw" text which can be static, like a database and files, or a noncontinuous stream, like E-mail or a plain file on a hard disk, or even a speech stream. Typically, this is data provided in an application data hooking up to the framework. The input access **12** receives as its input the "raw" text output by the input pool **10** and outputs "raw" text to the framework **14**. This is the interfacing layer where the data is accessed, possibly prepared and then handed over to the framework. As a last step this involves calling the frameworks outside API functions. Currently, we have provided ready-to-use input access layers (in the form of macros and API layers) for several applications like Lotus Notes, Microsoft Visual Basic, Microsoft Word, Java and also command line text file access.

The information extraction framework **14** includes preprocessor modules **141** that receive as input the "raw" text from the input access **12** and output "cleaned" text, possibly with additional technical information. This can involve stripping of irrelevant pieces of text (like technical mail headers), filtering out special characters of tags or converting between different character sets. This is done inside the framework using flexible, configurable preprocessor modules, that can be extended by user built preprocessor libraries in exactly the same fashion user built extractors can be integrated.

The information extraction framework **14** comprises an information extraction engine **142** which, in turn, includes control conditions **143** for extractor modules **144**. Information extraction is accomplished with the extractor modules **144**. In the illustrated embodiment, there is a set of extraction sub-engines comprising a parser **1441**, a classification engine **1442**, and a helper engine **1443** which extracts information by using heuristic techniques. The extractors are interchangeable, and other extractors may be added. This is possible because they are built according to a common API wrapper which defines the input/output functions.

Generally speaking, an extractor takes two pieces of input data, the natural language text to be processed and the information representation for the so far extracted information (which is usually empty at the first step), and generates an output. The output provided is an enriched information representation (and possibly modified text).

It is also possible to combine several extractors. For example, one can use a parser for preprocessing and/or postprocessing the text before or after classification. This can be useful for extracting phrasal or semantic information needed to give a better classification. Developers can add new additional extractors without access to the source-code of the framework by creating a library that exports certain functions. To use such a custom-built extractor library just put its filename in the framework configuration file (this is a plain text .INI file).

On this level of processing, some information processing is to be done which extends the scope of what the extractors are doing in two ways. First, information delivered by the engines can be mapped to user supplied domain knowledge (e.g., augmenting personal name information in the input data with other personal information from a database). Second, information delivered by an engine (which mainly operate at the sentence level) can be integrated at the level of the whole text.

The next level of processing handles the calling-sequence of the sub-engines as well as general reasoning in the information structures. The control part consults its configuration setting to decide which sub-engine to call first. Additional calling of other sub-engines can follow depending on the configuration and also on the output of the first step.

To allow for flexible control we provide a control language with Boolean expressions. The conclusion part provides some general reasoning/inferencing function on the extracted information using artificial intelligence (AI) techniques.

The information extraction framework **14** employs tools and resources **16**. Each extractor and stage of processing requires task-specific tools to create the resources needed during the extraction process. In the illustrated embodiment, the tools and resources include a configuration tool **161**, a grammar workbench **162**, and classifier tools **163**.

The output of the information extraction engine **143** is processed text and information representation (feature structure) which input to the action section **145**. The data structure for information representation from the information extraction engine **143** represents simple classification facts (e.g., "this text is about sports") as well as complex information extracted from the text (e.g., the author, a deadline, etc.) or grammatical/semantic information (e.g., a parse tree). The action section **145** is a rule based and pattern action system that supports action selection in a flexible and extendible manner. An action consists not only in the choice of what to do, but also in some "arguments" with additional information needed to perform the actual action (e.g., action="store mail", arguments="folder XYZ"). The action section **145** is separated from the calling application because a specific action (e.g., "store mail") may be executed differently in different environments or mail systems.

The action execution platform **18** receives the selected action with arguments for the action from the action selection **145**. The output of this platform is application dependent; that is, whereas the selected action output from the action selection **145** is application independent, the output for the action execution platform **18** is dependent on the calling application (e.g., Lotus Notes, Microsoft Word, etc.). This is because action execution typically happens in the application that called the framework. This could be inside a graphic user interface (GUI) (e.g., Windows 95/NT) or "inside" an application like Lotus Notes. It could also just display or spool some output information to a screen, a file or a database.

Prior to describing FIG. **2**, three basic data structures representing documents are described including (i) Infostructure, (ii) Documentstructure and (iii) Collectionstructure. More specifically, each Infostructure represents component parts and information about those parts of a document, e.g., words, phrases, sentences, paragraphs, content of subject headers, etc. To be more specific, each Infostructure is essentially a record structure of attribute value pairs, where each value itself could be an Infostructure, i.e., Infostructures can be no recursive. The Documentstructure is a group of infostructures organized in record format with document attributes such as subject, body, paragraph, etc. Some linguistic information can only be determined at this level, such as the resolution of pronoun reference (e.g., "Jon walked into the room. He sat down and talked to Mary." Identification of "he" as "Jon" is a discourse level function and would be represented in the Documentstructure). The Collectionstructure represents structure of or information about a collection of documents, e.g., what words occur and their frequencies, resolution of pronoun references based on analysis of multiple documents, etc. Training of classifiers occurs at this level.

Referring now to FIG. **2**, the text analysis framework for information extraction according to the invention is shown. An application (e.g., Lotus Notes, Microsoft Word, etc.) **20** is provided with a library integration layer **201** which enables the application to make a call to the framework. The application **20** submits the documents to the framework for processing and the library integration layer **201** functions to facilitate integrating the framework into the application. The call is received and processed by the application API **22** which provides the interface between the information extraction engine and the calling application.

The information is then submitted to the Textbuffer+ empty or pre-filled Documentstructure (e.g., Textbuffer) **204**. The Textbuffer **204** holds the incoming text for processing plus any structured information submitted to framework (pre-filled Documentstructure). The text is then processed (e.g., preparation) by linguistic analysis modules, e.g., document separation, language identification, tokenization, morphological analysis, etc. The linguistic analysis modules are referred to as services or service modules, since they are quite general. After each service module processes a document, the information gathered is recorded as Infostructures in the Documentstructure **206** representing all the information known about the current document. Thus, the Documentstructure is non-empty (partially filled).

The Partially Filled Documentstructure **206** provides a record of all information gathered at any point in the processing phase, which is driven by a Configuration File **[208] 210**. In fact, each step in the processing is determined by the Control Conditions **207**, which is driven by declaratively stated information in the Configuration file **[208] 210**. Thus, the Configuration File **[208] 210** dictates which modules are invoked, the sequencing, and indicates all other necessary parameters for each module. Also, the Configuration File **[208] 210** acts as a repository of information determining the flow of control of the Service, Extractor and Aggregator modules together with pointers to any information needed to drive the selected and sequenced modules. The information then goes through an aggregation process and submitted to the Collectionstructure **208**, which is a repository for all information gathered about groups of documents.

FIG. **2** is modular and provides various APIs for accessing modules such as Services API **212**, Error API **214**, Configuration API **216**, Display API **220**, Aggregators API **222** and Extractors API **224**. The Services API **212** access modules such as pre-processors, document separators, tokenizers, dictionaries and morphological analyzers/stemmers, language identifiers, etc. The Configuration API **216** drives the system based on parameters. The Extractors API **224** access modules such as classifiers, parsers and semantic interpreters, names and terms identifiers, etc. The Aggregators API **222** access modules that process collections of documents and the APIs (internal) functions are provided to communicate with plug-and-play components. It is well understood by one of ordinary skill in the art that the example Services, Extractor, Aggregator modules (plug-and-play), etc. merely illustrate some of the modules envisaged to be typically part of various applications. The framework of the present invention, however, is completely neutral as to what modules are used in a particular application.

FIG. **3** is a block diagram of the object inclusion and flow of control in the information extraction (IE) framework and represents the logic call flow by object class modules (in the standard sense of object oriented programming methodology) which are in direct correspondence to the process shown in FIG. **2**. Specifically, FIG. **3** shows an external C API layer **301** and C API layer **302**. The C API layer **302** provides the mechanism to communicate with the plug-and-play extractors, represented by the class IE_ExtrernalExtractor- Base class.

From a processing point of view, there are five object classes shown in FIG. **3**. These classes include the class IE_Framework **304**, class IE_ExecutionModules **306**, the class IE_Processors **308**, the class IE_Engine **310** and the class IE_Extractor **312**. The class IE_Extractor **312** determines the constraints on extractors and the class IE_Engine **310** determines what can be done with the extractors. The class IE_Preprocessors **308** define the generic characteristics of the preprocessor or Service modules (see FIG. **2**).

The C API Layer **301** functions to initiate the framework, submit the documents and terminate the processing. Specifically, the document is first handed to the object class IE_Framework **304** (information extraction framework object class). Inside the IE_Framework **304** are **4** data structures including the class IE_Infostruct (infostructures data structure) **304***a*, class IE_Document (documentstructures) **304***b*, class IE_Collectionstructure **304***c* (collection structures) and class IE_FrameworkConfig (configuration file data structure) **304***d*. An instance of the class IE_Framework **304** invokes instances of the class IE_Execution Modules **306**, where the class IE_Execution Modules **306** calls instances of the class IE_Preprocessors **308**. The class IE_Preprocessors **308** perform the Services depicted in FIG. **2**. The instances of the class IE_Execution Modules **306** calls instances of the class IE_Engine **310**, where the class IE_Engine **310**, in turn, calls instances of the class IE_Extractors **312**, which covers Service modules, Aggregator modules as well as Extractor modules of FIG. **2**. Depending on the information in the Configuration File **210**, various extractors are called sequentially to process the documents as explained in FIG. **2**. Once finished, the information in the Documentstructure (or Collectionstructure) is accessible via an API or can be returned as is.

Adaptability is one of the key issues motivating this architecture. We have made it possible to adapt the framework to different tasks just by replacing or configuring some of its components. A major goal has been to enable an "ordinary" user to make most or all of the adaption. This means that adaptation can often be done without programming or grammar writing. Where adaptation by programming is unavoidable, a developer can create a extension DLL with a specified simple API.

The information extraction framework is implemented in IBM Visual Age C++ (primarily because of its good portability). The target platforms for the framework are preferably Windows 95/NT, AIX/UNIX and OS/2. However, the framework of the present invention is designed to be platform independent. The whole Engine is a DLL (shared library on AIX). All extractors are DLLs also. Preexisting modules (parser, classifier) written in C/C++ could be integrated via wrapper DLLs hiding their specific API.

API Details

Application (Outside API) (Simplified)

To interface to the framework, an application has to call mainly five functions:

- init (ConfigurationFilename) This will initialize the framework. It has to be called only once per session. The following functions can be called in a loop for each document the application wants to analyze.
- clearDocumentStruct ( ) This will clear the results of the last document. (This not only needed for the first document in the loop.)

run(data) This will extract information from your data and store in an DocumentStruct

retrieve results (There are various functions for accessing the resulting DocumentStructure.)

terminate ( ) This will clean up and free memory. Has to be called only once per session.

In further embodiments, the application may call other functions, for example:

begindocument: ( )

runDocumentPart (docPartContent, docPartLabel)

enddocument ( )

Extractor (Inside API) (Simplified)

init (ConfigurationInformation) This will be called from the framework once per session to initialize the Extractor. The following functions can be called in a loop for each document the application gives the framework to analyze.

run (data) This will be called from the framework for each document the application wants to analyze. The extractor should process the data and store the results in the resulting InfoStructure. (There is a simple API for this.)

terminate ( ) This will be called from the framework once per session to clean up and free memory.

Configuration File Format

The configuration file for the framework is a simple text file in the standard (Windows). INI format. This means it consist of sections (marked be square brackets) that have entries with (possibly comma separated sequences) values. Comments are preceded by a semicolon.

An example configuration file for the framework looks like this:

;global settings [IE]

;output in GUI messageboxes (other values are CONSOLE and OFF) DisplayMode=GUI

;should the framework automatically display the results?DisplayResultingInfostruct=NO

;list of all preprocessor modules that the framework should call

;(this must be filenames of libraries/DLLs in the path) [Preprocessors] 1=StripHeader 2=DoubleFirstLine

;configuration information for (one) preprocessor [StripHeader] StripTill=Subject:, Subj:, Subject, Subj

;list of all preprocessor modules that the framework should call

;(this must be filenames of Libraries/DLLs in the path)

;can be followed by a label to describe the section with

;configuration information for this classifier [Extractors]

;External Extractor Classifier Yorktown with Reuters Data 1=IEExClaY, Reuters

;External Extractor Parser with Deadline Grammar 2=IEExPars, Deadline

;configuration information for the Classifier [IEExClaY Reuters] Rulefile= "D:\PROJECTS\IE\IEExClaY.dat\Reuters\ rules.txt" StopWords= "D:\PROJECTS\IE\IEExClaY.dat\Reuters\ Stopwds.txt" Dictionary= "D:\PROJECTS\IE\IEExClaY.dat\Reuters\ base12k.txt" Logfile= "D:\PROJECTS\IE\IEExClaY.dat\Reuters\ rulapply-.log" StoreAttribute=Categories UseLabels=NO

;configuration information for the Parser [IEExPars Deadline] Grammar= d:\projects\IE\IEExPars.dat\deadline\ deadline AttributeIfSucceeds=Categories ValueIfSucceeds= Deadline SentenceAttribute=Sentence ParseOnlyIfContains=deadline, due, delivered, ready, finished, finish TreatAsAbbreviations=info, etc, Conf

Header Files

```
Application (outside API)
/*-------------------------------------------------------------------------*/
/* Definitions */
/*-------------------------------------------------------------------------*/
#define IESESSIONHANDLE void*
/*-------------------------------------------------------------------------*/
/* Basic functions */
/* */
/*-------------------------------------------------------------------------*/
// this has to be the first call to the engine, an
// application has to store the SESSIONHANDLE and
// pass with each subsequent call
// IE_Init is to be called only once per session
// (IE_Run can be called multiple times)
// INIFILE is the path to an standard .INI file with
// the configuration for the engine
// RETURN: OK or Errorcode
extern StdCall DllExport IESESSIONHANDLE
    IE_Init(char* iniFile) ;
// passes text DATA data to the engine for
// processing
// results can be retrieved with the access
// functions below
// RETURN: OK or Errorcode
extern StdCall DllExport int IE_Run(IESESSIONHANDLE
    session, char* data) ;
// like IE_Run but with the path to a text file in
// FILENAME for processing
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_RunOnFile(IESESSIONHANDLE session, char*
    filename) ;
// should be called before/after each run/access to
// clear results
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_ClearInfostructs(IESESSIONHANDLE session) ;
// call this once per session to free memory
// (SESSIONHANDLE will be void after)
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_Terminate(IESESSIONHANDLE session) ;
/*-------------------------------------------------------------------------*/
/* Configuration functions */
/* */
/*-------------------------------------------------------------------------*/
// set the configuration to the path of a new or
// changed configuration file
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_SetConfiguration(IESESSIONHANDLE session,
    char* iniFile) ;
// get the path of the current configuration file to
// a string buffer
// RETURN: OK or Errorcode if string is too small
extern StdCall DllExport int
    IE_GetConfiguration(IESESSIONHANDLE session,
    char *buffer, int buffSize) ;
/*-------------------------------------------------------------------------*/
/* Result access functions */
/* */
/* these function provide access to the resulting
infostructure. */
/* the structure is a sequence of attribute value
pairs. */
/* each attribute is unique and has a "SEQUENCE" */
/* of values. */
/* each value in the sequence (starting with 1) */
/* can be either atomic (number, string etc.), */
/* or recursively a complete infostructure in */
```

-continued

Header Files

```
/* itself. */
/* examination of nested infostructures is done by */
/* setting the examination context to the */
/* infostructure in question. */
/* Most function provide access to the attributes */
/* in two ways: */
/* either by name of the attribute (S-postfix) */
/* or by the position of the attribute in the */
/* sequence of attributes (I-postfix). */
/* (this can be useful in FOR loops) */
/*-----------------------------------------------------------------------*/
// get a string representation of the resulting
// infostructure in bracketed notation (just for
// display purposes)
// RETURN: OK or Errorcode if string is to small
extern StdCall DllExport int
    IE_GetResultAsString(IESESSIONHANDLE session,
    char *buffer, int buffSize) ;
// get the number of attributes in the examinated
// infostructure
// RETURN: Number of Attributes
extern StdCall DllExport int
    IE_NumberOfResultAttributes(IESESSIONHANDLE
    session) ;
// get info if the value number VALPOS of the
// attribute with name ATTRNAME is atomic or not
// RETURN: True or False
extern StdCall DllExport int
    IE_IsResultAttrValueAtomicS(IESESSIONHANDLE
    session, char *attrName, const int valPos) ;
// get info if the value number VALPOS of the
// attribute with the position ATTRNBR is atomic or
// not
// RETURN: True or False
extern StdCall DllExport int
    IE_IsResultAttrValueAtomicI(IESESSIONHANDLE
    session, int attrNbr, const int valPos) ;
// get the number of values of the attribute with
// the name ATTRNAME
// RETURN: number of values, or 0 if ATTRNAME is
// not defined
extern StdCall DllExport int
    IE_NumberOfResultValuesAttrS(IESESSIONHANDLE
    session, char *attrName) ;
// get the number of values of the attribute with
// the position ATTRNBR
// RETURN: number of values, or 0 if ATTRNBR is out
// of range
extern StdCall DllExport int
    IE_NumberOfResultValuesAttrI(IESESSIONHANDLE
    session, int attrNbr) ;
// copy the string representation of the value at
// VALPOS of atribute ATTRNAME to a string buffer
// if valPos is 0, a comma separated list of the
// representations of all values is returned
// RETURN: OK or Errorcode if string too small
// NULL string if ATTRNAME is not defined, or VALPOS
// is out of range
extern StdCall DllExport int
    IE_GetResultAttrValueS(IESESSIONHANDLE session,
    char *attrName, int valPos, char *buffer, int
    buffSize) ;
// copy the string representation of the value at
// VALPOS of attribute ATTRNBR to a string buffer
// if valPos is 0, a comma separated list of the
// representations of all values is returned
// RETURN: OK or Errorcode if string too small
// NULL string if ATTRNAME is not defined, or VALPOS
// is out of range
extern StdCall DllExport int
    IE_GetResultAttrValueI(IESESSIONHANDLE session,
    int attrNbr, int valPos, char *buffer, int
    buffSize) ;
// append the new string value STRVALUE to the
// sequence of values for attribute with name
// ATTRIBUTE (creates a new attribute is ATTRIBUTE
// doesn't exist yet)
```

-continued

Header Files

```
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_AddValueToAttrS(IESESSIONHANDLE session,
    char *attribute, char *strValue) ;
// append the new integer value INTVALUE to the
// sequence of values for attribute with name
// ATTRIBUTE (creates a new attribute is ATTRIBUTE
// doesn't exist yet)
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_AddValueToAttrI(IESESSIONHANDLE session,
    char *attribute, int intValue) ;
/*-----------------------------------------------------------------------*/
/* The following functions are for use by */
/* extractors mainly */
/*-----------------------------------------------------------------------*/
/*-----------------------------------------------------------------------*/
/* Classifer functions */
/*-----------------------------------------------------------------------*/
// add a category (and optional a confidence
// measurement) to the current result set
// if CONFIDENCE is <= 0 it is ignored.
// the attribute label is determined by the engine
// RETURN: OK or Errorcode
extern StdCall DllExport int
    IE_AddCategory(IESESSIONHANDLE session, char
    *category, float confidence) ;
/*-----------------------------------------------------------------------*/
/* Display functions */
/* */
/* Extractors should use this functions to */
/* display information and errors. */
/* They will be shown according to the value of */
/* the INI variable Outputmode (GUI, Console, */
/* File, none) */
/*-----------------------------------------------------------------------*/
// get the current display mode (GUI, cosole, file,
// off)
// RETURN: current display mode (codes defined in
// display.hpp)
extern StdCall DllExport int
    IE_GetDisplayMode(IESESSIONHANDLE session) ;
// display a message
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_DisplayMsg(IESESSIONHANDLE session, char*
    msg) ;
// display an error
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_DisplayError(IESESSIONHANDLE session, char*
    msg) ;
// display/hide a progressbar returns immediately
// after showing/destroying the bar
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_DisplayProgress(IESESSIONHANDLE session, int
    showIt, char* title) ;
// set the level of the progressbar to LEVEL
// (ignored if LEVEL is smaller than previous one)
// level must be between 0 and 100
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_SetProgressLevel(IESESSIONHANDLE session,
    int level) ;
// add LEVEL to the value of the previous level
// level must be between 0 and 100
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_AddProgressLevelI(IESESSIONHANDLE session,
    int level) ;
// add LEVEL to the value of the previous level
// (float for more precicion)
// level must be between 0 and 100
// RETURN: OK or Errorcode
extern StdCall DllExport void
    IE_AddProgressLevelF(IESESSIONHANDLE session,
```

-continued

Header Files

```
    float level) ;
/*-------------------------------------------------------------------------*/
/* Configuration query/INI file functions */
/* */
/* Extractors should use this functions to query */
/* information about configuration values in the */
/* common .INI file for the engine and all */
/* extractors. */
/* INI files have sections that contain */
/* attributes with (possibly) comma separated */
/* sequences of) values. */
/* They have to pass the ID String they got */
/* passed during their initialisation as a first */
/* argument (SECTION) */
/*-------------------------------------------------------------------------*/
// copy the string representation of the value at
// VALPOS of atribute ATTRNAME to a string buffer
// if valPos is 0, a comma separated list of all
// values is returned
// RETURN: OK or Errorcode if string to small
// NULL string if SECTION of ATTRIBUTE is not
// defined, or VALPOS is out of range
extern StdCall DllExport int
    IE_GetIniValue(IESESSIONHANDLE session, char*
    section, char* attribute, int valPos, char
    *buffer, int buffSize) ;
// return the boolean value of the value at VALPOS
// of atribute ATTRNAME
// RETURN: true if the value is one of the
// following: TRUE, ON, YES, T, Y,+, 1)
extern StdCall DllExport int
    IE_GetIniValueAsBoolean(IESESSIONHANDLE
    session, char* section, char* attribute, int
    valPos) ;
// return the integer value of the value of VALPOS
// of atribute ATTRNAME
// RETURN: conversion of the value to int (0 if not
// successful)
extern StdCall DllExport int
    IE_GetIniValueAsInt(IESESSIONHANDLE session,
    char* section, char* attribute, int valPos) ;
// RETURN: number of attributes in section SECTION
extern StdCall DllExport int
    IE_NumberOfAttributes(IESESSIONHANDLE session,
    char* section) ;
// RETURN: number of values of attribute ATTRIBUTE
// in section SECTION
extern StdCall DllExport int
    IE_NumberOfValues(IESESSIONHANDLE session,
    char* section, char* attribute) ;
Extractor (inside API)
/*-------------------------------------------------------------------------*/
/* Definitions */
/*-------------------------------------------------------------------------*/
// just a handle definition for readability
#define IEEXTRACTORHANDLE void*
/*-------------------------------------------------------------------------*/
/* Exported functions */
/* these are all the functions an extractor */
/* Library has to export */
/*-------------------------------------------------------------------------*/
extern StdCall DllExport IEEXTRACTORHANDLE
    init(IESESSIONHANDLE sessionHandle, char*
    idString) ;
extern StdCall DllExport int run(IEEXTRACTORHANDLE
    extractorHandle, char* data) ;
extern StdCall DllExport int
    terminate(IEEXTRACTORHANDLE extractorHandle) ;
extern StdCall DllExport int
    getType(IEEXTRACTORHANDLE extractorHandle) ;
extern StdCall DllExport char*
    getInfo(IEEXTRACTORHANDLE extractorHandle, int
    infoType) ;
```

While the invention has been described in terms of a single preferred embodiment, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is as follows:

1. An information extraction architecture capable of extracting information from natural language documents, comprising:

application program interfacing means for receiving a natural language document from an application program and converting the natural language document into raw data, said interfacing means being configurable without altering a source code of said application program;

interfacing extraction means for receiving the raw data from the application program interfacing means and providing the raw data to an extractor, whereby the extractor extracts text and information from the raw data; and

action means for providing an application independent external action on the extracted data and outputting the application independent external action extracted data to the application program.

2. The architecture of claim **1**, further comprising:

initializing means for initializing a framework in response to the application program;

storing means for storing the extracted information; and

retrieving means for retrieving the stored extracted information.

3. The architecture of claim **2**, further comprising terminating means for terminating the extraction processes and clearing free memory.

4. The architecture of claim **1**, wherein the action means includes:

action execution means for executing a desired action, the action execution means being application dependent.

5. The architecture of claim **1**, further comprising

storing means for storing incoming text data received from the application program interfacing means prior to the text data being extracted and processed.

6. The architecture of claim **1**, wherein the application program interfacing means is an input access layer that is capable of interfacing with application programs.

7. The architecture of claim **1**, wherein the interfacing extraction means includes preprocessing means for cleaning the raw data.

8. The architecture of claim **7**, wherein the preprocessing means includes at least one (i) stripping means for stripping irrelevant pieces of text, (ii) filter means for filtering out special characters of tags and (iii) converting means for converting between different character sets.

9. The architecture of claim **1**, further comprising controlling means for controlling the interfacing extractor means during selection of a desired extractor.

10. The architecture of claim **1**, further comprising recording means for providing a record of all information gathered during extraction of the raw data.

11. The architecture of claim **1**, further comprising library means associated with the application program interfacing means, the library means providing a library of application programs so that the application program interfacing means interfaces between the application programs and the interfacing extracting means.

12. The architecture of claim 1, wherein the application program interfacing means is modular.

13. A method of extracting information from natural language documents, comprising:

interfacing an application program with a framework for information extraction from natural language documents, said framework being configurable without altering a source code of the application program;

interfacing extracted information from raw data, received in response to interfacing the application program with an extractor, the extracted information being processed text and information representation; and

providing an application independent action specification, associated with the extracted information, and outputting the application independent action specification to the application program for performing an application dependent implementation of the action specification.

* * * * *

US 20030105589A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0105589 A1

Liu et al. (43) Pub. Date: Jun. 5, 2003

(54) MEDIA AGENT

(76) Inventors: Wen-Yin Liu, Beijing (CN); Hong-Jiang Zhang, Beijing (CN); Zheng Chen, Beijing (CN)

Correspondence Address:
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

(21) Appl. No.: 09/998,092

(22) Filed: Nov. 30, 2001

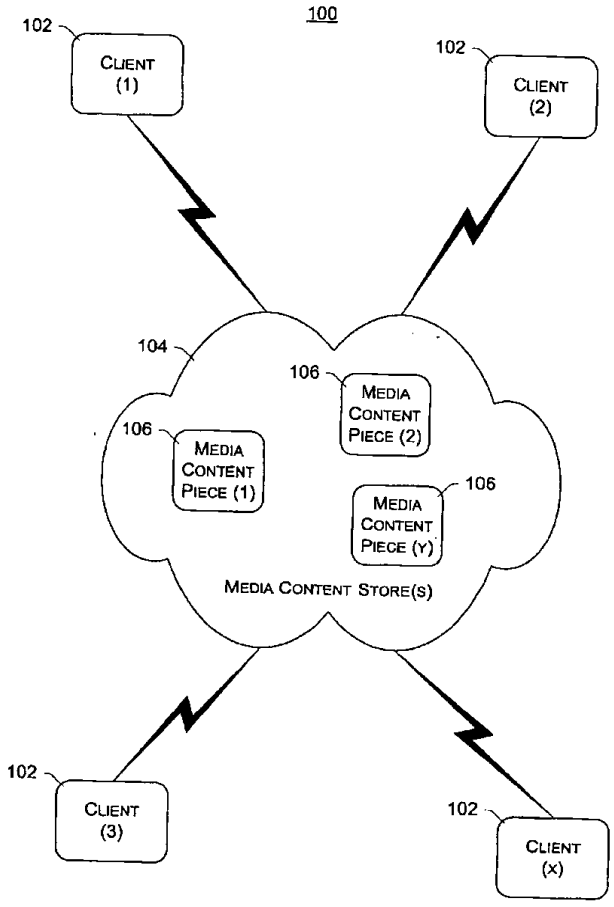
Publication Classification

(51) Int. Cl.[7] G06F 19/00

(52) U.S. Cl. 702/1

(57) ABSTRACT

The described arrangements and procedures provide an intelligent media agent to autonomously collect semantic multimedia data text descriptions on behalf of a user whenever and wherever the user accesses media content. The media agent analyzes these semantic multimedia data text descriptions in view of user behavior patterns and actions to assist the user in identifying multimedia content and related information that is appropriate to the context within which the user is operating or working. For instance, the media agent detects insertion of text and analyzes the inserted text. Based on the analysis, the agent predicts whether a user intends to access media content. If so, the agent retrieves information corresponding to media content from a media content source and presents the information to a user as a suggestion.



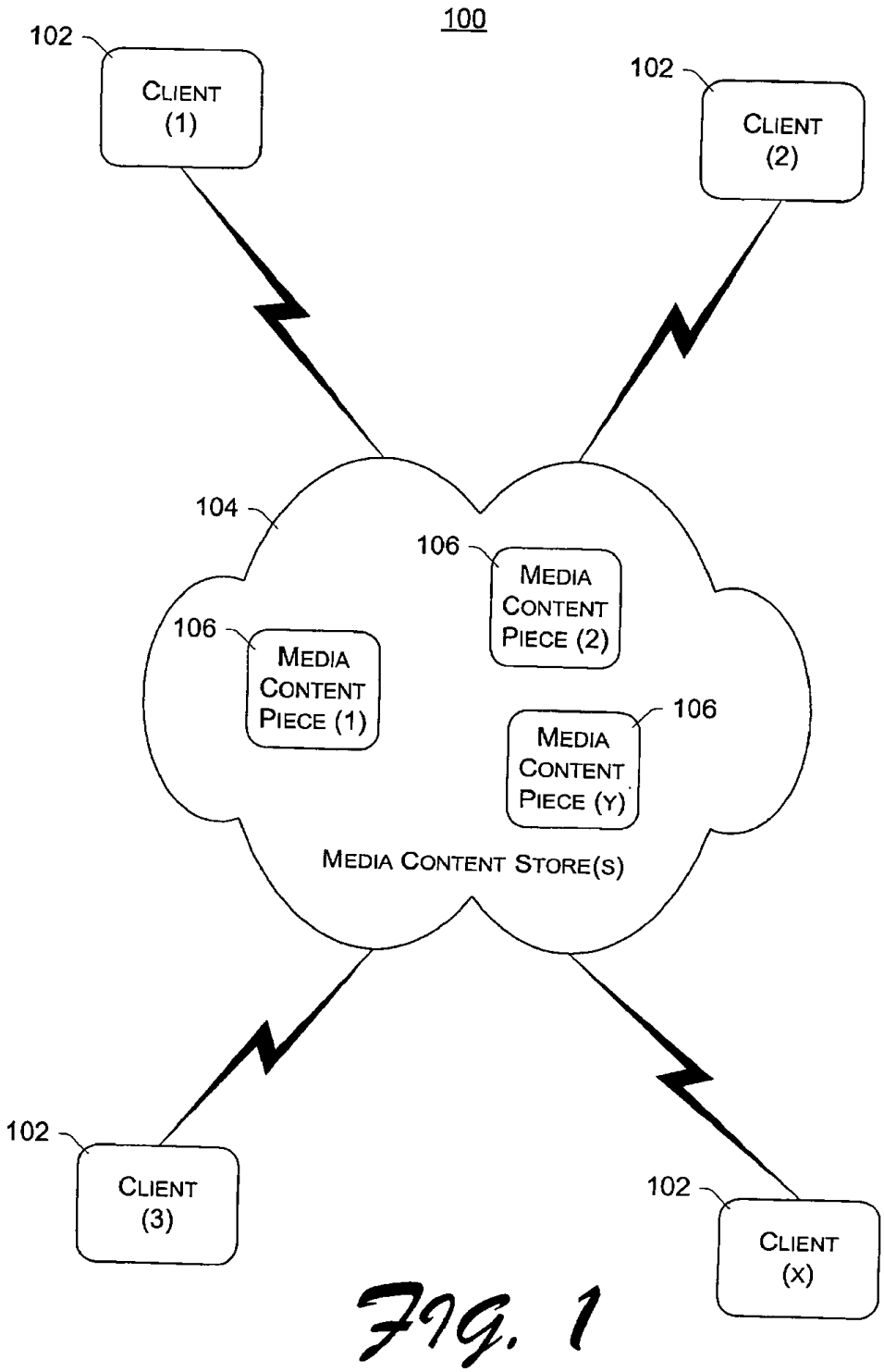

100
102
CLIENT (1)
102
CLIENT (2)
104
106
MEDIA CONTENT PIECE (2)
106
MEDIA CONTENT PIECE (1)
106
MEDIA CONTENT PIECE (Y)
MEDIA CONTENT STORE(S)
102
CLIENT (3)
102
CLIENT (X)

FIG. 1

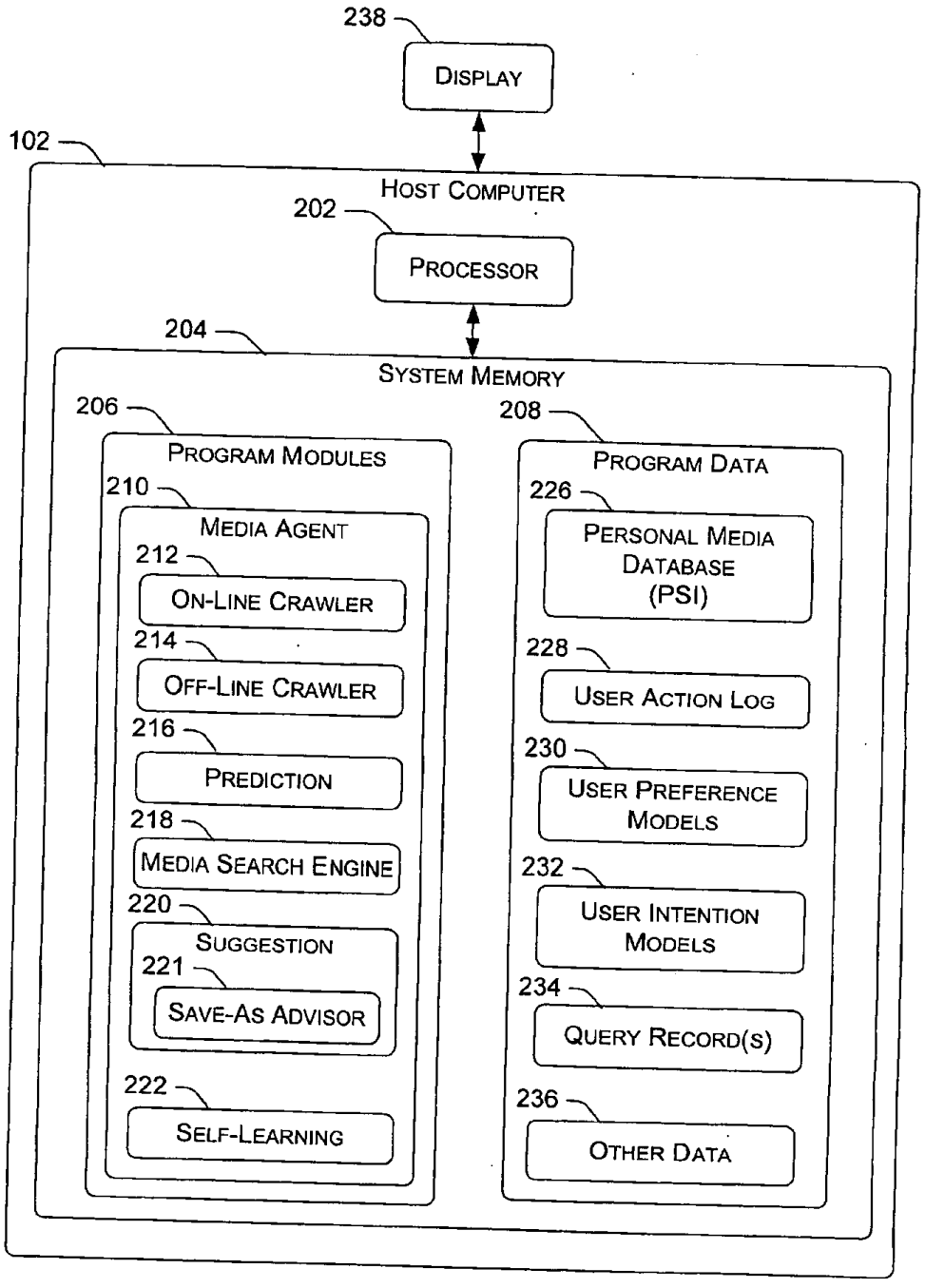

238
DISPLAY
102
HOST COMPUTER
202
PROCESSOR
204
SYSTEM MEMORY
206
PROGRAM MODULES
210
MEDIA AGENT
212
ON-LINE CRAWLER
214
OFF-LINE CRAWLER
216
PREDICTION
218
MEDIA SEARCH ENGINE
220
SUGGESTION
221
SAVE-AS ADVISOR
222
SELF-LEARNING
208
PROGRAM DATA
226
PERSONAL MEDIA DATABASE (PSI)
228
USER ACTION LOG
230
USER PREFERENCE MODELS
232
USER INTENTION MODELS
234
QUERY RECORD(S)
236
OTHER DATA

Fig. 2

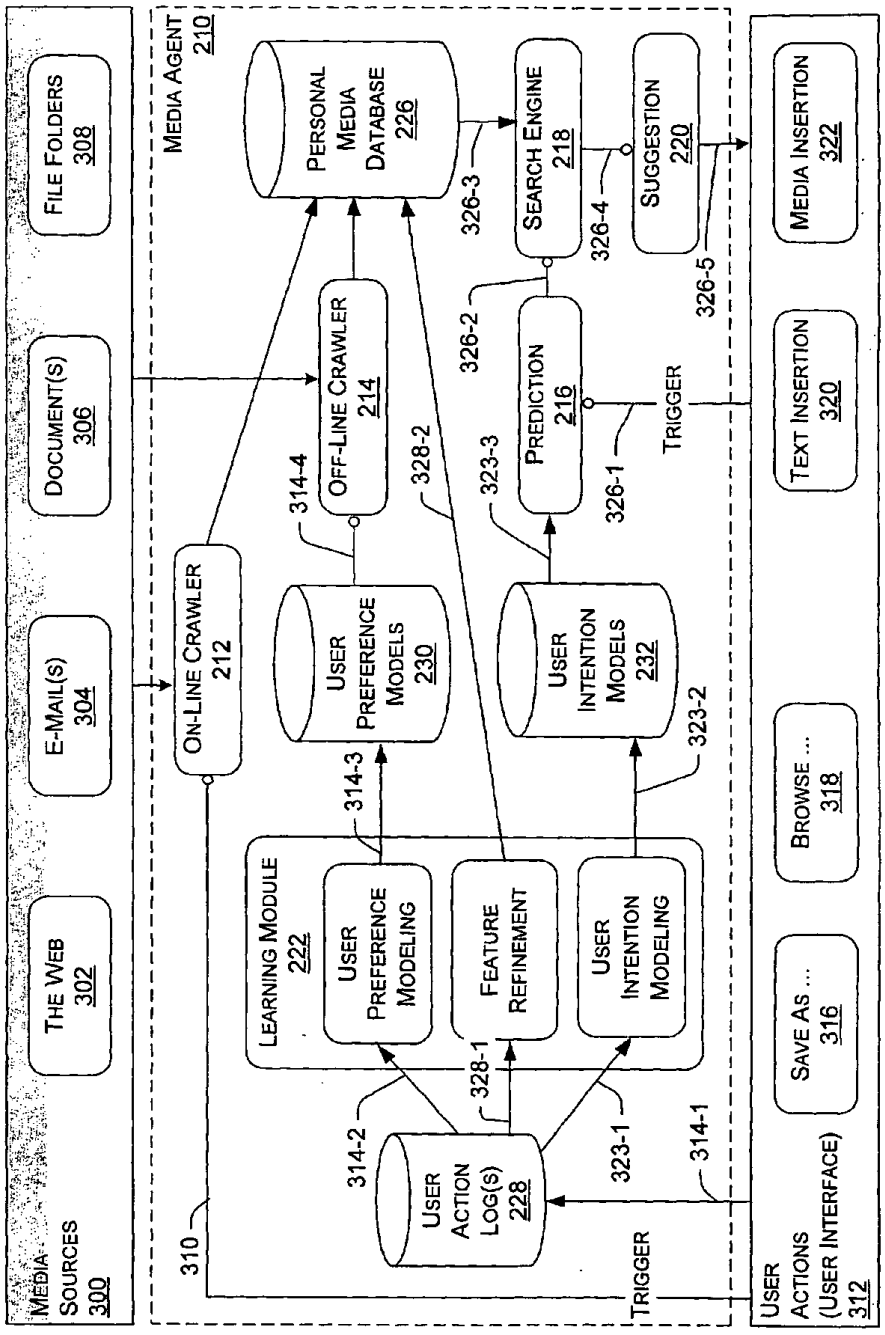

MEDIA SOURCES 300
THE WEB 302
E-MAIL(S) 304
DOCUMENT(S) 306
FILE FOLDERS 308
MEDIA AGENT 210
ON-LINE CRAWLER 212
OFF-LINE CRAWLER 214
SEARCH ENGINE 218
PREDICTION 216
SUGGESTION 220
LEARNING MODULE 222
USER PREFERENCE MODELING
FEATURE REFINEMENT
USER INTENTION MODELING
USER ACTION LOG(S) 228
PERSONAL MEDIA DATABASE 226
USER PREFERENCE MODELS 230
USER INTENTION MODELS 232
USER ACTIONS (USER INTERFACE) 312
SAVE AS ... 316
BROWSE ... 318
TEXT INSERTION 320
MEDIA INSERTION 322
TRIGGER
310
314-1
314-2
314-3
314-4
323-1
323-2
323-3
326-1
326-2
326-3
326-4
326-5
328-1
328-2

Fig. 3

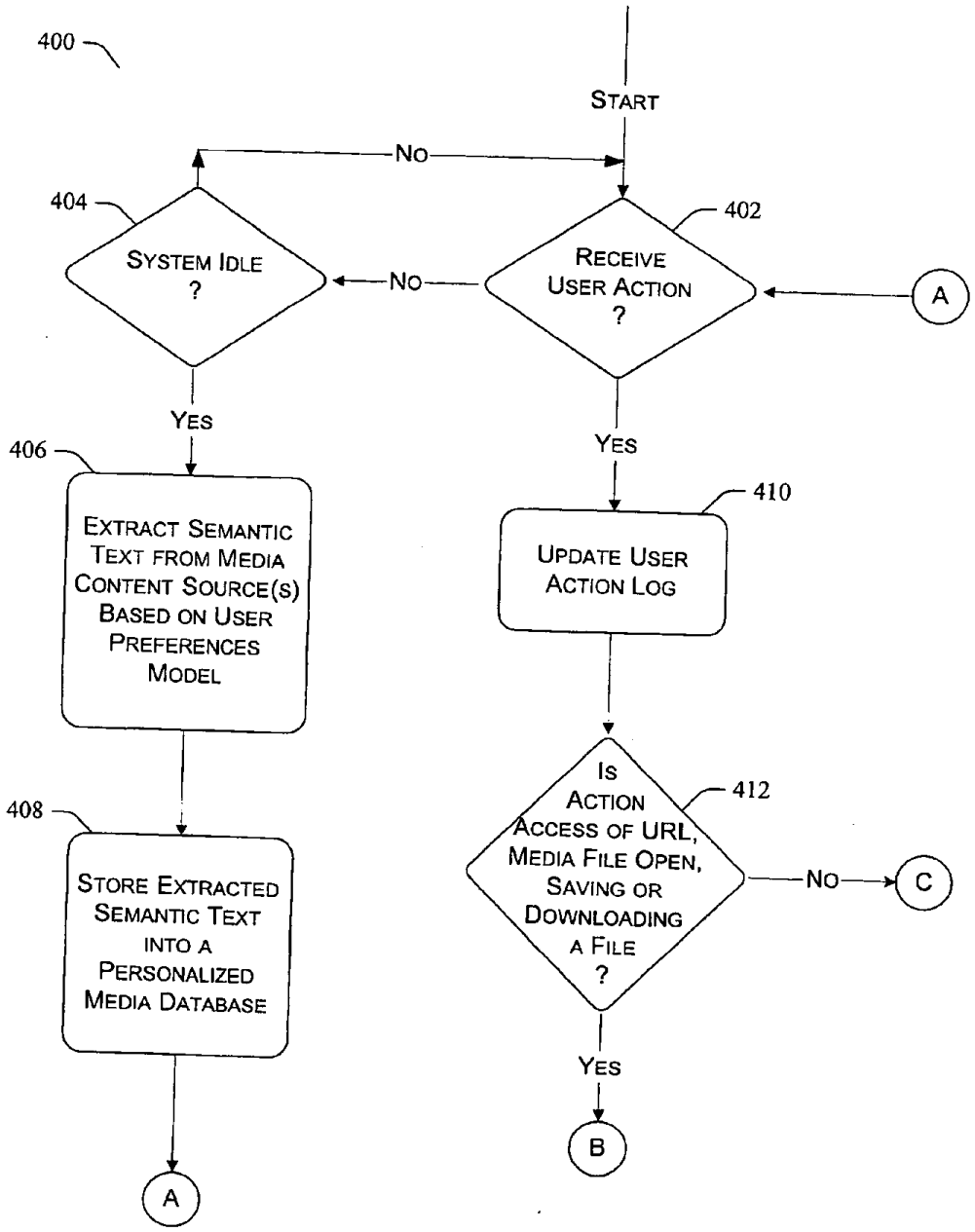

400
START
No
404
SYSTEM IDLE ?
No
402
RECEIVE USER ACTION ?
A
Yes
Yes
406
EXTRACT SEMANTIC TEXT FROM MEDIA CONTENT SOURCE(S) BASED ON USER PREFERENCES MODEL
410
UPDATE USER ACTION LOG
408
STORE EXTRACTED SEMANTIC TEXT INTO A PERSONALIZED MEDIA DATABASE
IS ACTION ACCESS OF URL, MEDIA FILE OPEN, SAVING OR DOWNLOADING A FILE ?
412
No
C
Yes
B
A

FIG. 4

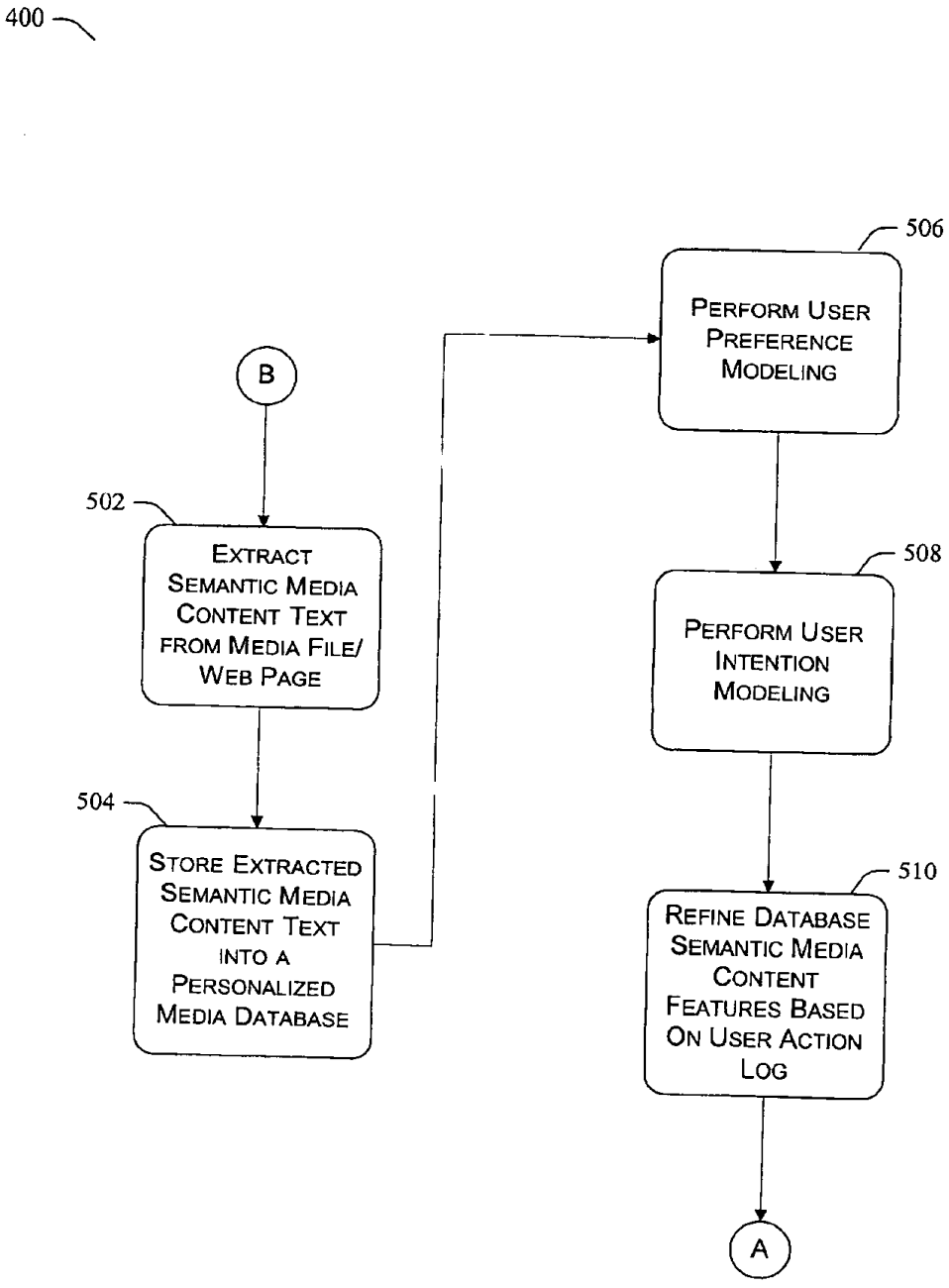

400
506
PERFORM USER PREFERENCE MODELING
B
502
EXTRACT SEMANTIC MEDIA CONTENT TEXT FROM MEDIA FILE/ WEB PAGE
508
PERFORM USER INTENTION MODELING
504
STORE EXTRACTED SEMANTIC MEDIA CONTENT TEXT INTO A PERSONALIZED MEDIA DATABASE
510
REFINE DATABASE SEMANTIC MEDIA CONTENT FEATURES BASED ON USER ACTION LOG
A

FIG. 5

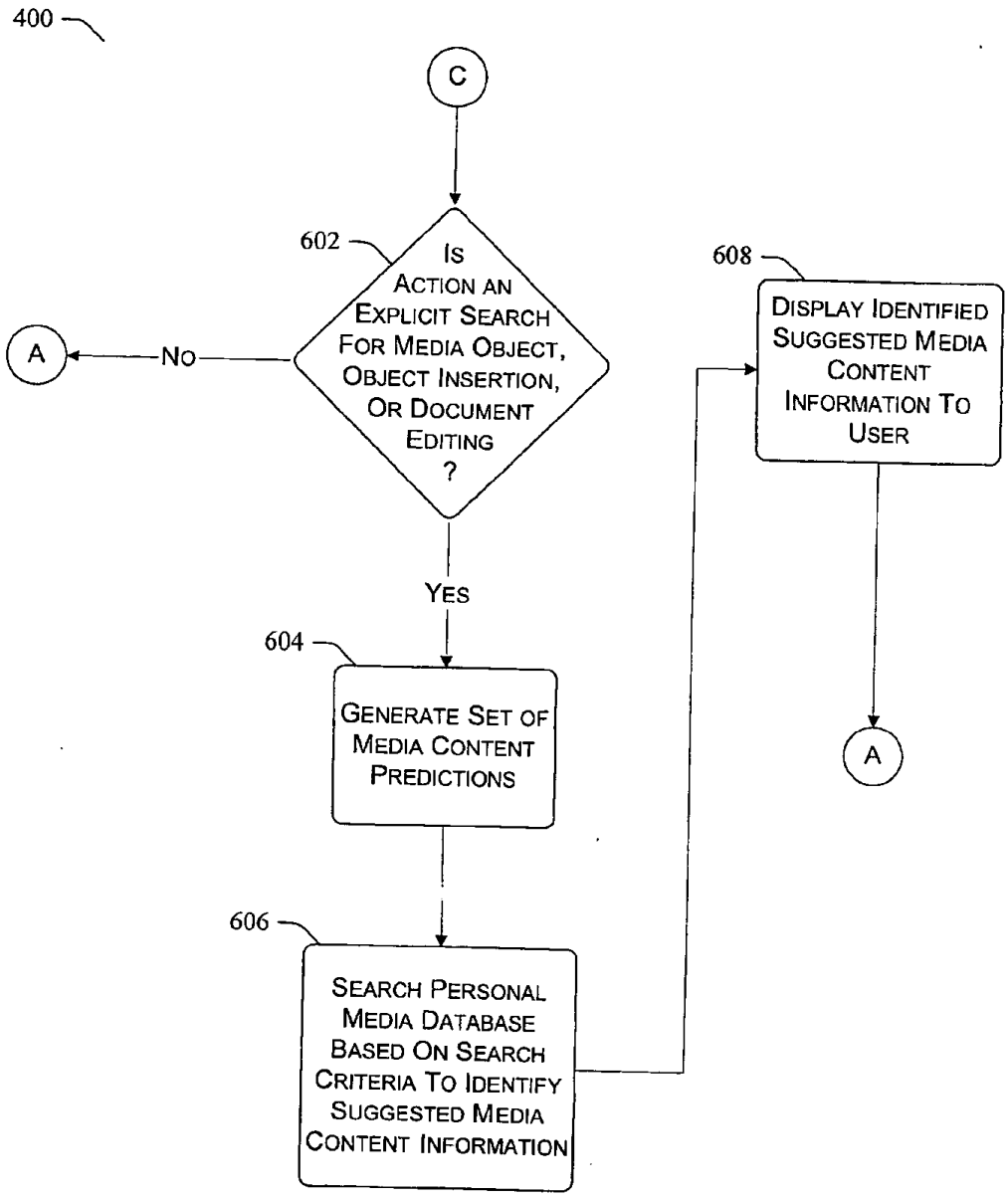

400
C
602
Is Action an Explicit Search For Media Object, Object Insertion, Or Document Editing ?
No
A
Yes
604
Generate Set of Media Content Predictions
606
Search Personal Media Database Based On Search Criteria To Identify Suggested Media Content Information
608
Display Identified Suggested Media Content Information To User
A

FIG. 6

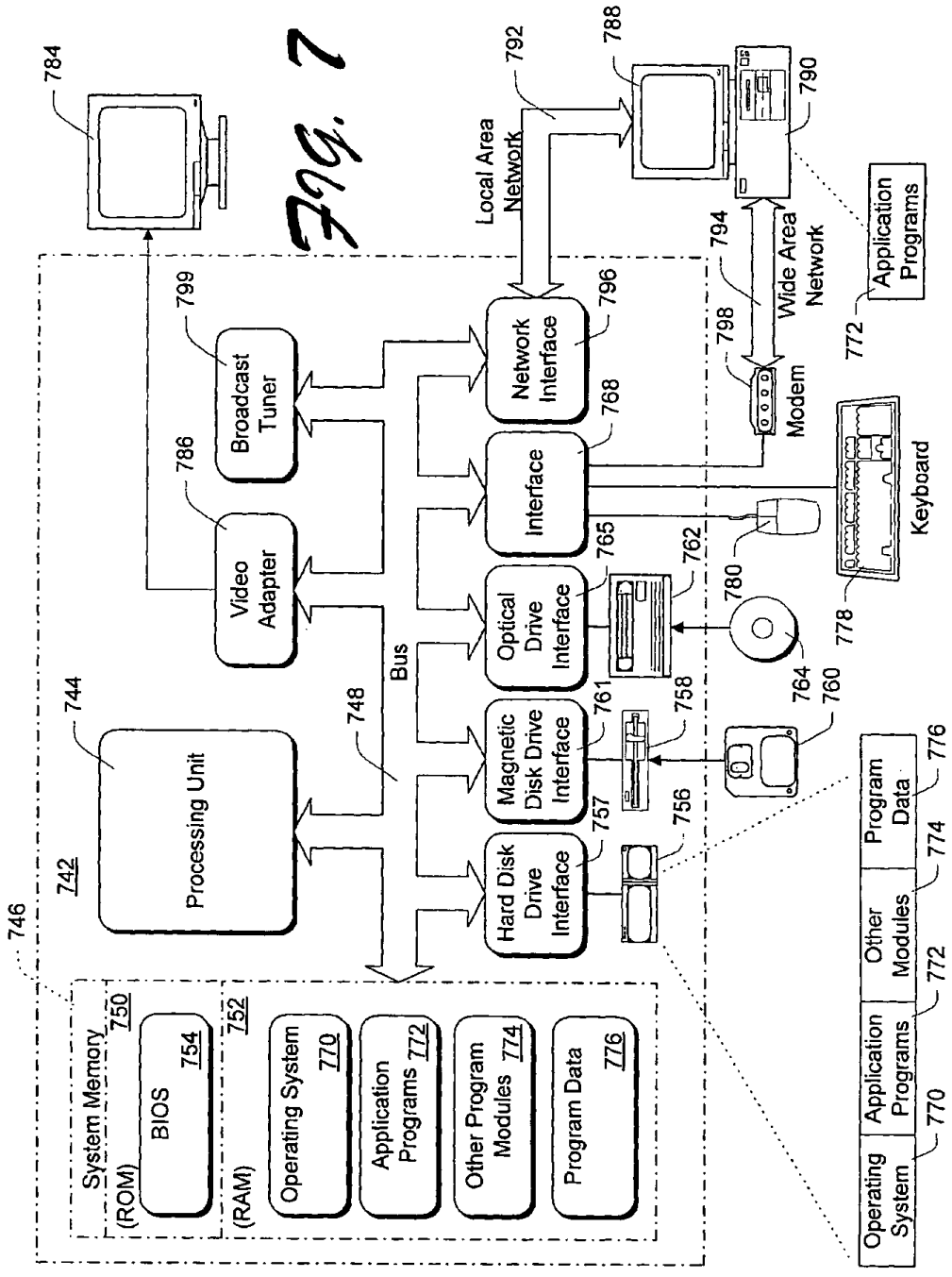


Fig. 7
784
792
788
790
Local Area Network
794
Wide Area Network
Application Programs
772
799
Broadcast Tuner
796
Network Interface
798
Modem
786
Video Adapter
768
Interface
765
762
780
Keyboard
778
Optical Drive Interface
764
760
761
758
Magnetic Disk Drive Interface
757
756
Hard Disk Drive Interface
744
Processing Unit
742
748
Bus
746
System Memory
750
(ROM)
BIOS
754
752
(RAM)
Operating System
770
Application Programs
772
Other Program Modules
774
Program Data
776
Operating System
Application Programs
Other Modules
Program Data

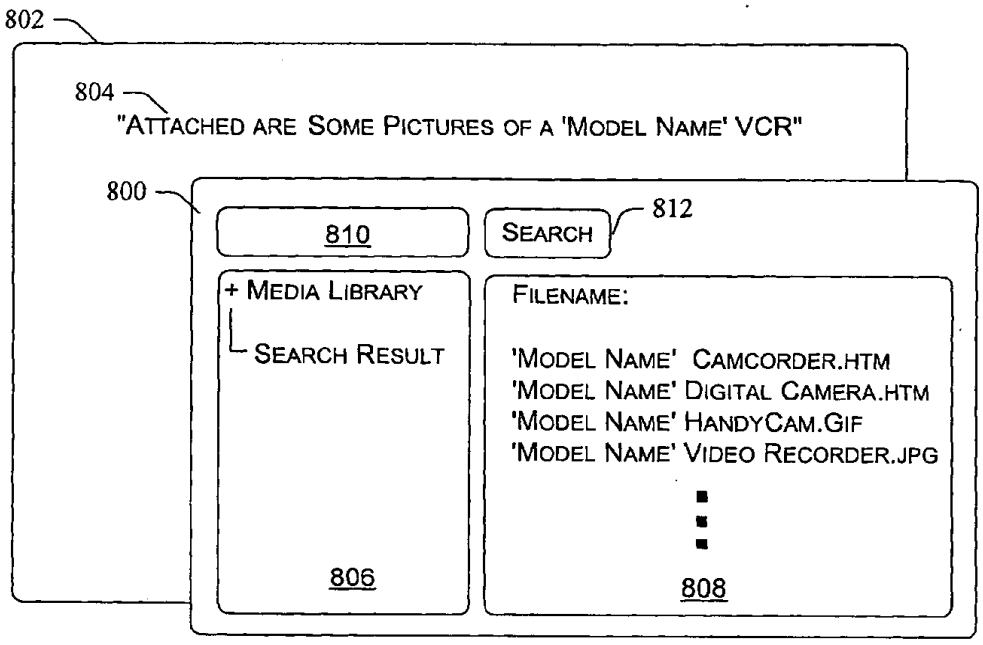

802
804
"ATTACHED ARE SOME PICTURES OF A 'MODEL NAME' VCR"
800
810
SEARCH
812
+ MEDIA LIBRARY
SEARCH RESULT
806
FILENAME:
'MODEL NAME' CAMCORDER.HTM
'MODEL NAME' DIGITAL CAMERA.HTM
'MODEL NAME' HANDYCAM.GIF
'MODEL NAME' VIDEO RECORDER.JPG
808

FIG. 8

MEDIA AGENT

TECHNICAL FIELD

[0001] The following description relates to use of multimedia.

BACKGROUND

[0002] The number of images and other types of media content that are available to users via their computers, especially with the evolvement of the Internet, has become very large and is continuing to grow daily. For instance, people often download media content such as multimedia files, images, videos, audio, and so on from the World Wide Web (WWW). Additionally, a number of known computer programs simplify user generation of personalized media files. Moreover, multimedia files are often used to enhance documents and are typically distributed via e-mail as attachments.

[0003] It is very difficult to manage and utilize large and dynamic sets of media content or multimedia data (e.g., media from a web page, an email attachment, a multimedia generation tool, and so on.) once it is accessed or saved into a user's computing environment. For instance, once such data are saved into local folders, substantial numbers of accumulated multimedia files are typically never used again because they are difficult for the user to locate (e.g., through a search). This is often the case because media files themselves may be stored in an ad-hoc manner.

[0004] One conventional technique to facilitate a user's explicit search for media content requires the manual annotation of media content to identify semantics of the media. This technique is substantially limited for a number of reasons. One problem with this conventional technique to identify image semantics is that an image must be manually annotated prior to the user's actual search for media content corresponding to the image. Another problem with this technique is that manually annotating multimedia to include text is a tedious process that is prone to human subjectivity and error. In other words, what one person may consider to be semantically related (e.g., the subject matter, pertinent, interesting, significant, and so on) to a particular image may be quite different from what another person may consider to be semantically related to the particular image.

[0005] Another conventional technique to facilitate a user's explicit search for media content analyzes text on a Web page text to identify semantics of images displayed on the page. This analyzed text is compared to the user's search query. If it matches to some extent, then the Web page may include media that is related to the user's search. This technique is substantially limited in that images on a Web page may have semantics other than what is specifically recited with the text on the Web page.

[0006] The following arrangements and procedures address these and other problems of managing and accessing multimedia data.

SUMMARY

[0007] The described arrangements and procedures provide a media agent to detect and analyze inserted text. Based on the analysis, the media agent predicts or anticipates whether a user intends to access media content. If so, the media agent retrieves information corresponding to the anticipated media content from a media content source. The media agent presents the retrieved media content based information to the user as a suggestion.

[0008] Responsive to user access of a media content source, the media agent collects media content and associated text from the accessed media content source. Semantic text features are extracted from the media content and the associated text. These semantic text features are indexed along the collected media content into a media database that may be personalized for the user.

[0009] The media agent monitors a user's actions to determine the user's media content use preferences. For instance, when the user's computer system is in an idle state (e.g., when the processor is not 100% active and has unused processing cycles), the agent collects media content and associated text from a media content source. Such an idle state may occur at any time, for instance, when a user is typing an e-mail message, and so on. The agent extracts semantic text features from the media content and the associated text. The agent determines that the media content is of interest to the user based at least in part on semantic similarity between the media content use preferences and the semantic text features. If the media agent determines that the media content is of interest to the user, the agent indexes the semantic text features into the user's personal media database.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The same numbers are used throughout the drawings to reference like features and components.

[0011] **FIG. 1** illustrates an exemplary environment in which the invention can be practiced.

[0012] **FIG. 2** shows an exemplary host computer to semantically index, suggest, and retrieve media content according to personal usage patterns.

[0013] **FIG. 3** shows exemplary aspects of process and data flows between modules and data sinks in the media agent module. Specifically, **FIG. 3** shows sequences in which data transfer, use, and transformation are performed during the execution of the media agent.

[0014] **FIG. 4** shows an exemplary procedure to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 4** shows a procedure to determine whether offline gathering of media content semantics, online gathering of media content semantics, preference and intention modeling, or user intention prediction and suggestion procedures should be performed.

[0015] **FIG. 5** shows further aspects of an exemplary procedure to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 5** shows further aspects of a procedure for a media agent of **FIGS. 2 and 3** to perform online gathering of media content semantics and preference and intention modeling.

[0016] **FIG. 6** shows further aspects of exemplary procedures to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 6** shows further aspects of a proce-

dure for a media agent of **FIGS. 2 and 3** to determine whether preference and intention modeling or user intention prediction and suggestion procedures should be performed.

[0017] **FIG. 7** illustrates aspects of an exemplary suitable operating environment in which a media agent to semantically index, suggest, and retrieve media content according to personal usage patterns may be implemented.

[0018] **FIG. 8** shows an exemplary user interface to present media suggestions (e.g., filenames) for a user to select based on what the user has typed into a window such as an e-mail message or other document.

DETAILED DESCRIPTION

[0019] The following description sets forth exemplary subject for a media agent to semantically index, suggest, and retrieve media content and other information corresponding to media content according to a user's personal media use patterns. The subject matter is described with specificity to meet statutory requirements. However, the description itself is not intended to limit the scope of this patent. Rather, the inventors have contemplated that the claimed subject matter might also be embodied in other ways, to include different elements or combinations of elements similar to the ones described in this document, in conjunction with other present or future technologies.

[0020] Overview

[0021] As discussed in the background section, using conventional techniques it is very difficult to manage and utilize large and dynamic sets of media content once it is accessed or saved into a user's computing environment because media files themselves may be stored in an ad-hoc manner. However, this is also the case because locating a particular multimedia file that is relevant to a context within which the user is working (i.e., the user's intent) is a substantially difficult problem. And, unless the user is performing an explicit search for media content, none of the described conventional procedures take into account multimedia content needs of the user within the context that he or she is working.

[0022] In contrast to such conventional procedures, the following arrangements and procedures provide for an intelligent media agent to autonomously collect semantic multimedia data text descriptions on behalf of a user whenever and wherever the user accesses multimedia data. The media agent analyzes these semantic multimedia data text descriptions in view of user behavior patterns and actions to assist the user in identifying multimedia content that is appropriate to the context within which the user is operating or working. To accomplish this, the media agent provides timely prompts with suggested multimedia content and/or information corresponding to media content (e.g., suggested media filenames).

[0023] **FIG. 1** illustrates an exemplary environment to identify a context within which the user or client is working and suggest semantically related multimedia content for the client to work with based on the identified context. In environment **100** one or more (x) clients **102** are coupled to a media content store **104**. The media content store **104** is any combination of local storage (e.g., local volatile or non-volatile memory), networked storage (e.g., a parallel connection, an organizational intranet network, the Internet, and so on), or other communication configurations.

[0024] These communication configurations provide for electronic exchange of information using an appropriate protocol (e.g., TCP/IP, UDP, SOAP, etc.) between the host device **102** and one or more media content sources or servers that include multiple (y) pieces of media content **106**. This electronic exchange provides for client **102** communication with media content store **104** to access (e.g., view, search, download, etc.) pieces of media content **106**.

[0025] The storage of media content pieces **106** within media content store **104** can be arranged in any of a wide variety of manners and according to any of a wide variety of data formats. For example, media content pieces **106** may be stored on multiple servers hosting Web pages accessible via a network using an appropriate protocol such as Hypertext Transfer Protocol (HTTP). Web pages are documents that a user can view or otherwise render and which typically include links to one or more other pages that the user can access. Web pages are typically stored as one or more files at a remote location(s), being accessed by the user via a computer that is operatively coupled to a network. Web pages often include multiple pieces of media content **106**.

[0026] Media content pieces **106** include any of a wide variety of conventional media content, such as audio content, video content (for example, still images or frames of motion video), multimedia content, etc. A piece of media content **106** refers to media content that can be rendered, such as a single visual image, an audio clip (e.g., a song or portion of a song), a multimedia clip (e.g., an audio/video program or portion of an audio/video program), etc. The described arrangements and procedures can be used with a wide variety of conventional media content.

[0027] In the illustrated example, a user of a client **102** accesses the media content store **104** for pieces of media content **106**. The client **102** automatically detects a user's access or utilization of a media object **106** (e.g., an image, a chart, an audio, a video, an Excel® file, etc.) and collects semantic text descriptions of the accessed media object **106** during its use. These media object descriptions are extracted from text associated with an accessed media content piece **106**.

[0028] Media content **106** may coexist with corresponding text description. The source of the text description may be part of the multimedia content itself or can be independent but semantic to the content. For instance, an e-mail message may describe attached media content (e.g., an attached image or video clip). Many other composite documents, including slide files, word processor documents, and so on, also commonly include both media content and corresponding text. All of these files can be used as potential sources of semantic features of media data. Thus, the client **102** collects or extracts semantic features of a media file from text descriptions from the media content's environment (e.g., the Web page, the e-mail, the compound document, and so on).

[0029] As a user operates within the computing environment of a client **102**, the client **102** monitors the user's activities and provides suggestions of semantically related media content **106** to use based on these user activities in view of the collected media object descriptions. For instance, after authoring a paragraph of text description

during preparation of a technical report, a user indicates that he or she wants to insert some multimedia content **106** (e.g., a diagram). (There are any numbers of ways for the user to author such a paragraph such as via user input with a keyboard, a stylus, a mouse or other pointing device, voice recognition, and so on). The client **102** anticipates the desire to insert the content and/or the particular content that the user wishes to insert. This is accomplished by mapping information from surrounding text description (e.g., text above, below, or to the sides of the media content insertion point) to user prediction and preference patterns and stored multimedia data text descriptions. Using this information, a list of one or more anticipated multimedia items is presented (e.g., in a dialog box) to the user for user selection. An example of this is described in greater detail below in reference to **FIG. 8**.

[0030] An Exemplary System

[0031] **FIG. 2** shows an exemplary host computer **102** to semantically index, suggest, and retrieve media content according to personal usage patterns. The host computer **102** is operational as any one of a number of different computing devices such as a personal computer, an image server computer, a thin client, a thick client, a hand-held or laptop device, a multiprocessor system, a microprocessor-based system, a set top box, programmable consumer electronics, a wireless phone, an application specific integrated circuit (ASIC), a network PC, minicomputer, mainframe computer, and so on.

[0032] The host computer includes a processor **202** that is coupled to a system memory **204**. The system memory **204** includes any combination of volatile and non-volatile computer-readable media for reading and writing. Volatile computer-readable media includes, for example, random access memory (RAM). Non-volatile computer-readable media includes, for example, read only memory (ROM), magnetic media such as a hard-disk, an optical disk drive, a floppy diskette, a flash memory card, a CD-ROM, and so on.

[0033] The processor **202** is configured to fetch and execute computer program instructions from program modules **206**; and configured to fetch data **208** while executing the program modules **206**. Program modules typically include routines, programs, objects, components, data structures, etc., for performing particular tasks or implementing particular abstract data types. For instance, program modules **206** include the media agent module **210**, and other applications (e.g., an operating system, a Web browser application, and so on.

[0034] The media agent module **210** includes on-line crawler **212** and off-line crawler **214** modules, a prediction module **216**, a media search engine **218**, a suggestion module **220**, and a self-learning module **222**, each of which are described in greater detail below. The media agent module **210** automatically detects user actions with respect to media content to trigger one or more appropriate modules **212** through **222**. Media content (e.g., Web pages, composite documents that include media content such as e-mails, word processing files, and so on) refers to any one or more of the media content pieces **106** of **FIG. 1** and/or media represented in a user's personal media database **226**.. Actions with respect to media content include, for example: accessing a URL (e.g., with respect to a media content piece **106**), creating a media object, importing or downloading a media object, inserting a media object (e.g., into a document), opening, saving, updating or editing a media object, exporting or uploading a media object **106**, and so on.

[0035] The Online and Offline Media Content Crawler Components

[0036] The online **212** and offline **214** crawler modules are triggered at various times to: (a) collect potentially related high-level features (also referred to herein as semantic text features) of a media object from a composite document (e.g., an e-mail, Web page, or word processing document); (b) extract semantic text features from the media object itself; and (c) index the media object in the personal media database **226** using the collected and extracted semantic text. A composite document includes both media content and corresponding text (e.g., an e-mail message with an attached picture or slide file, word processor documents, etc.). For instance, if the composite document is an e-mail with an attachment, the crawlers **212** and **214** may extract semantic text features (e.g., words) from both the body of the e-mail message and from the attached media content piece itself.

[0037] Specific actions that trigger the on-line crawler module **212** include, for example: visiting a URL, saving/downloading a media object from the Web or an email, saving a media hyperlink from the Web, inserting a media object or its link into a document or an e-mail, and so on. The off-line crawler module **214** is activated at system **102** idle time to collect and index semantic text corresponding to media objects local or remote to the host **102** (e.g., the media content pieces **106** of **FIG. 1**) that are similar to the user's preferences models **230**. (User preferences models **230** are described in greater detail below).

[0038] Media content semantic text features are extracted by crawlers **212** and **214** in a variety of different manners. For instance, text features are extracted based on up to six aspects of the text associated with media content: (1) a filename and identifier, (2) an annotation, (3) alternate text, (4) surrounding text, (5) a page title, and/or (6) other information. Note that all of these aspects may not be associated with each media content piece, and thus features are not extracted based on aspects that are not available for the media content piece.

[0039] (1) Image filename and identifier: each image is identified by a filename that is typically part of a larger identifier that indicates where the file is located (e.g., a URL). Often meaningful names are used as filenames and/or the identifier (e.g., URL) for an image. Each word in the filename and identifier can be used as a text feature. In one implementation, a set of rules is used to judge the usefulness of the filenames and URL for an image, and thereby limit the words used as text features.

[0040] One rule is that the filename be segmented into meaningful key words. Based on a standard dictionary (or alternatively a specialized dictionary), the filename is analyzed to determine whether it includes one or more words that are in the dictionary. Each such word is identified as a key word. For example, the filename "redflower.jpg" would be segmented into the key words "red" and "flower", each of which would be a text feature (assuming they each existed in the dictionary).

[0041] Another rule or criteria is that certain common words (e.g., articles) are excluded from being considered

key words. For example, the filename "theredflower.jpg" could be segmented into the words "the", "red", and "flower", but only "red" and "flower" would be text features (the word "the" is a stop-word and thus not identified as a key word). Other insignificant characters and groups of characters can also be excluded, such as digits, hyphens, other punctuation marks, filename extensions, and so on.

[0042] Another rule applies to the URL for an image. A URL typically represents the hierarchy information of the image. The URL is parsed and segmented to identify each word in the URL, and then resulting meaningful key words are used as text features. For example, in the URL ". . . /images/animals/anim_birds.jpg", the words "animals" and "birds" are meaningful key words that would be extracted as images. A dictionary can be used to identify the meaningful key words as discussed above. For example, the word "images" would not be meaningful as only images are being analyzed.

[0043] (2) Image annotation: each image can have a corresponding image annotation which is a text label describing the semantics of the image, typically input by the creator of the image file. This image annotation is intended to describe the semantics of the image. Thus, each word in the image annotation may be a key feature (although certain common words and/or insignificant characters/character groups can be excluded as discussed above regarding image filenames and identifiers).

[0044] (3) Alternate text: many web pages include alternate text for images. This alternate text is to be displayed in place of the image in certain situations (e.g., for text-based browsers). As this alternate text is intended to replace the image, it often includes valuable information describing the image. Thus, each word in the alternate text is a key feature (although certain common words and/or insignificant characters/character groups may be excluded as discussed above regarding image filenames and identifiers).

[0045] (4) Surrounding text: many web pages have text surrounding the images on the rendered web page. This text frequently enhances the media content that the web page designers are trying to present, and thus is frequently valuable information describing the image. Thus, key words from the text surrounding the image (e.g., text above the image, below the image, to the left of the image, and to the right of the image) are extracted as text features (certain common words and/or insignificant characters/character groups may be excluded as discussed above regarding image filenames and identifiers). The amount of text surrounding an image from which key words are extracted can vary. For instance, the three lines (or sentences) of text that are closest to (adjacent to) the image are used, or alternatively the entire paragraph closest to (adjacent to) the image can be used. Alternatively, if information is available regarding the layout of the web page, then the single sentence (or line) most related to the image can be used.

[0046] (5) Page title: many times a web page will have a title. If the web page does have a title, then key words are identified in the title and used as text features (certain common words and/or insignificant characters/character groups may be excluded as discussed above regarding image filenames and identifiers).

[0047] (6) Other information: other information from the web page may also be used to obtain words to be used as text features associated with an image. For example, each URL on the page that is a link to another web page may be parsed and segmented and meaningful key words extracted from the URL (analogous to the discussion above regarding extracting meaningful key words from the URL of the image). By way of another example, meaningful key words may be extracted from "anchor text" that corresponds to the image. Anchor text refers to text that is identified on the web page as text that should be kept near or next to the image (e.g., which would cause the browser to move the text to a next page if the image were to be displayed on the next page). Key words can be extracted from the anchor text analogous to the discussion above regarding extracting meaningful key words from the alternate text.

[0048] After applying these various rules, the crawler **212** or **214** has a set of words that are text features extracted from the image. Note that certain words may be extracted multiple times and thus appear in the set multiple times. The crawler module **212** or **214** stores these high-level semantic text features and an identifier of the media content piece (e.g., a URL) in personal media content and features database **226**. The media content piece itself may also optionally be stored in a separate database **236** from the high-level semantic text features.

[0049] The extracted high-level text features are a set of words. The crawler module **212** or **214** takes the extracted features for media content from personal media content database **226** and indexes the media content piece. These generated feature vectors or indices are stored in personal media content database **226** or alternatively elsewhere. The indexing process refers to generating, as necessary, feature vectors corresponding to the media content piece and storing a correlation between the generated feature vectors and the media content piece.

[0050] The crawler module **212** or **214** converts the extracted high-level text features into a text feature vector D_i for image i using a well-known TF*IDF method:

$$D_i = TF_i * IDF_i = \left(t_{i1} * \log\frac{N}{n_1}, \ldots, t_{ij} * \log\frac{N}{n_j}, \ldots, t_{im} * \log\frac{N}{n_m} \right) \quad (1)$$

[0051] where m represents the total number of different keywords maintained in database **226**, t_{ij} represents the frequency of keyword j appearing in the extracted set of words associated with image i, n_j represents the number of images identified in database **140** that contain the keyword j, and N represents the total number of images in database **140**. Each keyword in the text feature vector of an image is thus weighted based on how frequently it appears in the text associated with the image as well as how frequently it appears in the text associated with all images identified in database **226**. The resultant text feature vector D_i for image i thus includes a numerical element for each word that is in the text associated with at least one image identified in database **226** (if the word is not associated with image i then the value for that element is zero).

[0052] Each time new high-level semantic text feature vectors are added to database **228**, the previously generated feature vectors are re-generated. crawler modules **212** and **214** may generate (and re-generate) feature vectors based on

the features in database **226** as soon as new features are added to database **226**, or alternatively wait for multiple new features to be added to database **226**, or wait for a particular time (e.g., wait until early morning when fewer users will be accessing a computer's resources).

[0053] Accordingly, the personal media database **226** is personal to a particular user because it indexes all media objects that the particular user has accessed or accumulated from the digital world, including media content from the Web, the local machine **102**, and all other media content stores **104** such as e-mail and other composite documents. Once accumulated or otherwise accessed media content is indexed by semantic text features, text-based search of the media files is possible.

[0054] For instance, U.S. patent application Ser. No. 09/805,626 to Li et al., filed on Mar. 13, 2001, titled "A Media Content Search Engine Incorporating Text Content and User Log Mining", which is assigned to the assignee hereof and hereby incorporated by reference, describes searching a database using semantic text features of media content.

[0055] The User Prediction Component

[0056] The prediction module **216** monitors a user's typing actions and guesses or anticipates whether the user may want to insert a media object based on the user intention model **232**, which is described in greater detail below. To precisely predict the user' intention, the prediction module **216** generates the user intention model **232** based on a set of training data **236**. For instance, the user's intention can be modeled using a Bayesian Belief Network (BBN) to represent probabilistic relationships among three levels of semantic features: lexicography or "lexics", syntax, and patterns. BBNs are known tools to represent probabilistic relationships. The user intention modeling process is presented in greater detail below in reference to the learning module **222**.

[0057] The prediction module **216** uses the user intention model **232** and typed user text information to anticipate whether the user may want to insert a media file, and if so, the type of media file to insert. Specifically, the prediction module **216** extracts a set of keyword features from the text that the user has just typed and inputs the extracted keyword features to the BBN. The probabilities of all predefined user intentions are calculated based on the input keyword features and the one with the largest magnitude is chosen as the predicted user intention.

[0058] The prediction module **216** may determine or predict that a user desires to use/insert a media file into a document based on what a user types. This information can be used to predict even the context of the media file(s) that the user may wish to access. For instance, when the user is writing a document (e.g., an email), after the user types in text such as "The following are some pictures of digital cassette recorder", the prediction module **216** analyzes the text to guess that the user may want to insert some pictures of a digital cassette recorder, and therefore automatically activate the media search engine **218**, which is discussed in greater detail below, to locate media content that corresponds to digital cassette recorders.

[0059] **FIG. 8** shows an exemplary user interface **800** to present media suggestions (e.g., filenames) for a user to insert into a document **802** based on what a user has typed **804** into a window (e.g., an e-mail message). In this example, the user has typed text **804** into an e-mail application window **802**. The text **804** indicates that "Attached are some pictures of 'model name' VCR". The prediction module **216** analyzes this text **804** to guess that the user may want to "attach" some "pictures" of a video cassette recorder (i.e., "VCR") into the e-mail message **802**. Responsive to this guess, the media search engine **218** (discussed below) is activated to locate media content that corresponds to VCRs. Upon locating such corresponding media this information is presented (e.g., by the suggestion module **220**—which is discussed in greater detail below) to the in a window **800** (e.g., a media player window).

[0060] The media player window **800**, in this example, includes an area **806** to indicate that the search result has been incorporated into the media library or personal media database **226**. Window **808** indicates suggested media content (e.g., filenames) based on the user input text **804**. The user can simply select and drag and drop the suggested media content **808** into the document **802** if one or more of the suggestions are correct. The suggestions **808** can also be edited to more closely approximate or indicate desired content—or simply ignored by the user (e.g., a cancel button on a dialog box can be selected).

[0061] Windows **800**, and **806-812** represent only an example of a user interface with which to present suggested media content to a user based on the media agent's **210** determination that a user desires to insert media content into a document. For instance, there are a number of different ways for the media agent to detect the user's desire to insert media content. The user may select a drop-down menu item to indicate that an image is to be inserted at the current location in the document, information in text surrounding an insert point (e.g., "see the following diagram") may indicate that media content is to be inserted, and so on.

[0062] More details of the prediction process **216** are presented below in reference to the user intention model **232**.

[0063] The Search Engine Component

[0064] If it is determined that the user wants to access media content (e.g., to insert something into a composite document), the media agent **210** uses the media search engine **218** to locate relevant media objects based either on a search query that is explicitly specified by the user or automatically guessed by the prediction module **216**.

[0065] A user generates a search query by inputting a textual description of the search criteria pertaining to the types of media content desired. The textual description is then converted to a text feature vector and stored as a query vector **234**; otherwise the prediction module **216** has automatically generated the query vector **234** responsive to user actions (e.g., typing text).

[0066] A query vector **234** is generated by extracting keywords from search criteria (e.g., user input) and building the query vector (having the same number of elements as the semantic text feature vectors in database **226**, and each element corresponding to the same keyword as the corresponding element in the text feature vectors) by assigning a value of one to the element corresponding to each extracted keyword and a value of zero for the other elements. If an image is used for the search criteria, then keywords of any text description corresponding to that image are extracted and used to generate the initial high-level query vector. The keywords can be extracted in the same manner as discussed above with reference to online and offline crawler modules **212** and **214**.

[0067] The high-level query vector **234** is then generated by assigning a value of one to the element corresponding to each extracted keyword and a value of zero for all other elements. If the image retrieval process is initiated based on both an input text description and an input image, the high-level query vector is generated based on extracted keywords from both the input text and the input image. For example, initial vectors may be generated as discussed above (assigning a value of one to the element corresponding to each keyword), and then the vectors combined (e.g., elements added together or averaged on a per-element basis) to generate the initial high-level query vector **234**.

[0068] The search engine **218** uses a matching algorithm to determine the most relevant media objects that match to the user's intent represented by the generated query vector **234**. The matching algorithm calculates semantic similarity between the query vector **234** and each media object represented in the personal media database **226**. Semantic similarity is calculated using a dot product of the query's semantic feature vector **234** and the media object's semantic feature vector.

[0069] For instance, the similarity, referred to as $S_h(q_h, D_{i_h})$, between the high-level query vector q_h and the high-level feature vector of the image D_i, referred to as D_{i_h}, is calculated using the dot product of the query's text feature vector and the image's text feature vector as follows, which is a normalized similarity.

$$S_h(q_h, D_{i_h}) = \frac{q_h \cdot D_{i_h}}{|q_h||D_{i_h}|}.$$

[0070] The Suggestion Component

[0071] Once the search engine finds a set of relevant media objects in the personal media database **226**, the suggestion module **220** shows (e.g., via the display **238**) search engine **218** results to the user in a sorted list (e.g., in a dialog box) according to their semantic similarity to the query vector **234**. Each object is displayed with a short paragraph of text or a few keywords to describe its content. The user may select an item from the list for acceptance. For instance, the user may select a suggested item by double clicking it or by dragging and dropping one or more of the suggested items from the display into a document.

[0072] Additionally, if the user places a cursor over a suggested item such as a suggested media content item or filename, the suggestion module **220** may display all or a portion (e.g., keywords) of the semantic text stored in the personal media database **226** that corresponds to the suggested item. This additional information can be displayed in a number of different ways such as in a hovering window near the cursor hot-point, in a status bar, and so on. Moreover, the user may decide at this point to modify the semantic descriptions of the media objects in the database **226** to more particularly indicate the semantics of the media content.

[0073] Additionally, when a user wants to save or download a media object (e.g., a multimedia file, html file, audio file, video file, image file, and so on) from a media source such as from the Web or from an e-mail message, the suggestion module **220** can include a "save-as" advisor **221** to present one or more suggested filenames for the user to utilize to save or download the media object to the personalized media database **226**. These filenames are presented in a "Save-As" dialog box or window on the display **238** and are based on semantic features that are extracted from the media object and/or the media source. Such semantic features include, for example, filenames, surrounding text, page titles, hyperlinks, and so on. These semantic features are extracted from the media source by the on-line crawler **212**.

[0074] For instance the "save as" advisor **221** is activated when a user wants to save or download a media object from the Web (i.e., a Web page). The "Save As" Advisor automatically collects and extracts semantic information such as one or more keywords from the Web page. From these extracted keywords, the advisor suggests a list of corresponding filenames for a user to select. The user can modify or choose a suggested filename to use as the filename of the saved media object on the local machine **102**.

[0075] The Learning Component

[0076] Learning is a significant aspect of the media agent **210**. The media agent **210** improves performance based on relevance feedback from user interactions with the system. The users' interactions are recorded in the user action log **228**. The self-learning mechanism of the media agent **210** is implemented in a number of aspects, including: (a) learning to refine semantic features of accumulated media files; (b) learning user preferences models for automatically indexing non-visited but relevant media files; and (c) learning the user intention model **232** to provide more accurate suggestions to a user.

[0077] Responsive to user selection of one or more of the suggestion module **220** displayed suggestions, the learning module **222** automatically refines the semantic features of the search query **234** and updates the semantic indexing of the media objects in the personal media database **226**. To accomplish this, the learning module **222** accesses relevance feedback from the user log **228** and updates the query vectors to reflect the relevance feedback provided by the user. The query vector **234** is modified as follows:

$$Q' = Q + \beta\frac{\sum Q^+}{n^+} - \gamma\frac{\sum Q^-}{n^-}$$

[0078] where Q' represents the updated query vector, Q represents the original query vector, Q^+ represents the set of feature vectors of user selected media content, n^+ represents the number of user selected media content, Q^- represents the set of feature vectors of the non-selected media content, n^- represents the number of non-selected media content, β represents a weighting for positive feedback, and γ represents a weighting for negative feedback. Initially, the values of β and γ are set empirically, such as β=1.0 and γ=0.5. Alternatively, if some training data is available, the parameters can be tuned using the training data to improve the performance of the retrieval.

[0079] If a query vector **234** did not previously exist, then an initial query vector **234** is generated based on the relevance feedback. For example, feature vectors of the relevant images may be averaged together to generate a corresponding semantic text query vector to store in the personal media database **226**.

[0080] In this manner, suggested semantic features that result in positive user feedback are reinforced in the personal media database **226**. Additionally, by learning from the user's log **228** of whether the user accepts or rejects suggestions, the media agent **210** determines appropriate times to provide suggestions, potentially saving processing time (e.g., searches). Additionally, user habits can be determined

to anticipate when media content suggestions (i.e., provided by the suggestion module **220**) may or may not be desired. Additionally, frequently accessed media files usually show the user's preferences and profiles, which can be learned more precisely by recording user actions over a period of time. Once a user preference model **230** is known, the media agent **210** (i.e., the online or offline crawlers **212** and **214**) may automatically collect media objects pertaining to the user's interests from various media content sources **104**.

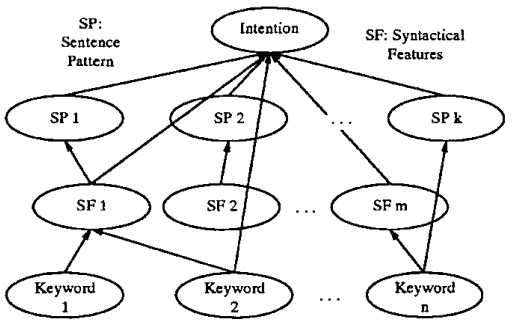
[**0081**] The User Intention Modeling Component

[**0082**] The self-learning mechanism **222** also includes user intention modeling **232** and preference modeling **230** based on the log **228** of a user's actions on accumulated media content. Many kinds of user activities, including mouse movement and typing can be used to learn and predict the user's intentions. For instance, when the user is writing a new e-mail and has typed "Here is an interesting picture download from the web", the probability of the user's intention of inserting an image into the e-mail body as an attachment is very high. Therefore, the media agent **210** (i.e., the prediction module **216**) can predict that user wants to insert an image in the e-mail. If the user's intention is to insert, the suggestion module **220** can provide potential images for the user to insert based on other text information the user has typed or will type.

[**0083**] All text factors that may imply the user's intentions are referred to as linguistic features. A Bayesian Belief Network is used to precisely represent the dependencies and probabilities among the linguistic features and the user's intentions. Three levels of linguistic features are defined: lexics, syntax, and a partial or completely instantiated sentence pattern. A lexical feature is a single word extracted from the text. A syntactical feature is the syntax structure of a sentence. An instantiated pattern feature is a frequently used sentence structure with some of its syntactical units instantiated with certain words or phrases, e.g., "Here it is a . . . " and "Attached please find . . . " The particular Bayesian Belief Network used to represent the user's intention model **232** is illustrated below in table 1.

TABLE 1

Example of User Intention Modeling using a Bayesian Belief Network



[**0084**] Initially, the user intention model of Table 1 is empty but is subsequently learned using the user's log **228** (a set of user action records) as training data. The user's log **228** records user action records to train the intention model. Each user action record contains a text part and a tag of whether a media file is attached. For instance, an e-mail message could have a text body and a media file attachment. The text part is parsed such that all words (lexical features) are extracted from the sentences and are stemmed.

[**0085**] At the lexical level, direct association between keyword features and user intentions is determined through training. A fast algorithm proposed by Agrawal et al. [1] can be used to generate rules for determining associations between keywords and intention(s). The rules represent the causality relationship between keywords and intentions.

[**0086**] For example, a set of rules identify whether there is a causal relationship between certain keywords and the intention to insert a media file. The causality rules are further constrained by two parameters: α (Support of Item Sets) and β (Confidence of Association Rule). The first parameter (α), which depicts a scope that the rules can be applied to, is expressed by the percentage of those records that contain the same keyword as evidence. The second parameter (β) depicts the probability that the rule stands, i.e., the probability of the intention given the appearance of the keyword. The generated rules are evaluated based on the values of these two parameters. The higher the two values, the better the rules. Those rules with parameters higher than certain thresholds (e.g., α=0.03, β=0.6) are selected to build the Bayesian Belief Network.

[**0087**] The Intention Prediction Process

[**0088**] Once direct associations between keyword features and user intentions are determined through training, the intention model **232** for a user can be used by the prediction module **216** to predict the user's intention based on what the user has just typed.

[**0089**] To accomplish this, a set of keyword features represented by ($<a_1, a_2 \ldots a_n>$) are extracted from the text typed by user. The prediction module **216** then calculates the probabilities of all predefined user intentions (V), and selects the intention with the biggest probability (v_{map}) using the following equation [11].

$$v_{map} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2, \ldots, a_n) = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, \ldots, a_n | v_j) P(v_j) \quad (2)$$

$$\text{where } P(a_1, a_2, \ldots, a_n | v_j) = \prod_{i=1}^{n} P(a_i | \text{Parents}(a_i), v_j).$$

[**0090**] In addition to lexical features, other informative features are used to precisely predict the user's intentions. For instance, natural language processing (NLP) technologies can be utilized to analyze sentence structures of the text. NLP can analyze a sentence and parse it into a tree structure. The highest-level sentence structures are utilized.

[**0091**] For example, "Here are some photos" is parsed into the following sentence structure: AVP ("here"), VERB ("be"), NP ("some photos"), wherein "AVP" represents an indication of an adverb phrase element, and "NP" represents a noun phrase element. Such syntactical features are used to determine additional useful information. This method [1] can also be used to generate association rules between these syntactical features and user intentions.

[0092] Use of syntactical features improves user intention prediction precision. However, certain sentence patterns, such as, "here is something" in an e-mail message typically indicates that a user intends to insert an attachment. The sentence structure is AVP+VERB+NP. Yet, the sentence "how are you" has the same structure and indicates a different intention. Therefore, parts of the sentence structure are further evaluated to locate instantiated patterns that may strongly indicate user's intentions. An instantiated pattern is a sentence structure specified with a pattern of words.

[0093] The rules generated at the lexical level using the method of [1] are substantially specific and lack complete representation of user intent. Hence, association rules are generated for instantiated pattern features based on the association rules found between syntactical features and user intentions. By instantiating parts of the syntactical features with certain frequently used words or phrases, association rules are generated at the instantiation pattern level, which is more general than the syntax level rules and more specific than the lexics level rules.

[0094] Since each syntactical unit can be replaced by many words, all combinations of words found in the training data and syntactical units are tested in a breadth-first order. Only those instantiated patterns that have α and β parameters (of the association rules) that are greater than certain thresholds are selected for user intention prediction.

[0095] The User Preferences Modeling Component

[0096] A user model includes many things about a user. The association rules and the intention models discussed above are part of the user intention model **232**. This section focuses on how to identify user interests and preferences (e.g., the user preference model **230**) from the user's interaction history with the media agent **210** as identified in the user action log **228**.

[0097] User preferences are modeled by analyzing semantic features of the media files that the user has accumulated and frequently used. By doing so, a scope of media contents matching user interest is identified. Once user preferences models **230** are identified, the media agent **210** provides appropriate suggestions (e.g., via the suggestion module **220**) or preferred media files automatically collected from all possible sources by the offline crawler **214**. Additionally, media files on the local machine **102** can be automatically and periodically sought for better indexing, clustering, and/or classification. Moreover, media files can be shared with other users that have similar user preferences models **230**.

[0098] To improve user preference modeling, several different preference models, each of which can be represented by a list of keywords, can be maintained for a user. For instance, all user log records **228** are clustered into several preferences clusters based on their semantic similarity. The semantic similarity for two keyword vectors is calculated using their dot product and normalized through the cosine method [5][13] (these methods were discussed above in reference to determining semantic similarity with respect to search engine **218** results). Each cluster corresponds to a preference model **230** for the user, which is represented by a keyword frequency vector formed by the top 10 frequently used keywords (except for stop words) and their frequency in the user log cluster. A user preference model is therefore represented by $m=<k_1, k_2, \ldots, k_{10}>$.

[0099] Whether a media file or object is of interest to the user also depends on the semantic similarity between the media object and one of the user preferences models **230**. The one with the largest similarity value (which is also large enough, e.g., larger than a threshold) is considered as relevant to the user's interest. Similarities between two user preference models **230** are compared by calculating the dot product of their keyword frequency vectors.

[0100] Modeling user's preferences based on keyword probability is another approach to determining keyword frequency in text documents. Specifically, the Naïve Bayes approach is used with respect to all words and their probabilities to form a keyword probability vector to model a preference [11]. The probability of word w_k is estimated using the following equation:

$$P(w_k|m_j) = \frac{n_k + 1}{n + |\text{Vocabulary}|}, \tag{3}$$

[0101] where n is the total number of words (or actually, the total length of text) existing within the training data, which are all user log records in the cluster corresponding to the user preference model m_j, n_k is the number of times that word w_k is found among these n words, and |Vocabulary| is the total number of distinct words found in the training data. In comparison, equation (2) is simply the term frequency combined with a smoothing function.

[0102] Given a multimedia document D represented by $<w_1, w_2, \ldots, w_n>$, the most probable user preference model M_{NB} is calculated using the Naïve Bayes approach as follows.

$$\begin{aligned} m_{NB} &= \underset{mj \in M}{\text{argmax}} P(m_j|w_1, w_2 \ldots w_n) \\ &= \underset{mj \in M}{\text{argmax}} P(w_1, w_2 \ldots w_n|m_j)P(m_j) \\ &= \underset{mj \in M}{\text{argmax}} P(m_j)\prod_k P(w_k|m_j) \end{aligned} \tag{4}$$

[0103] P_{mj} is the prior of m_j, which can be considered as of a uniform distribution initially. The approach assumes that the probability of a word is independent of others or its position within the text. Note that this assumption is not always true. However, in practice, the Naïve Bayesian learner performs remarkably well in many text classification problems despite this independence assumption [11].

[0104] $P(m_j|w_1, w_2 \ldots w_n)$ is comparable among different m_j, and can therefore be used to find a better model. However, $P(m_j|w_1, w_2 \ldots w_n)$ is not comparable among different D, since it differs in magnitude for different lengths of keyword vectors. To judge whether D is of the user's interest, another metric is required that is comparable among different D such that a value larger than a threshold means that the document is of the user's preference. First al all, due to multiple multiplications in equation (3), a geometrical mean is considered in normalizing as follows.

$$\sqrt[\log(n_w)]{P(m_{NB})\prod_i P(w_i|w_j))} = \frac{\log(P(m_{NB}|w_1, w_2 \ldots w_n))}{n_w},$$

[0105] where n_w is the number of distinct keywords in document D matched with keywords in the model M_{NB}.

Secondly, a factor of matched keyword percentage is considered such that the document containing a larger percentage of keywords in D matched in the model will get a higher metric value and is therefore more relevant to the user's preference model. Hence,

$$\frac{n_D}{n_w}$$

[0106] is multiplied, where n_D is the total number of words in D. Finally, the metric is defined as follows, which is used to measure the relevancy of D to the user's preference model.

$$P_{norm}(D) = \frac{n_D * \log(P(m_{NB}|w_1, w_2 \ldots w_n)}{n_w^2}, \quad (5)$$

[0107] Using the Bayesian method, the similarity between two user preference models is calculated. One of the metrics follows:

$$Sim(m_1, m_2) = \frac{P(m_1|m_2) + P(m_2|m_1)}{2}, \quad (6)$$

[0108] where, m_1 and m_2 are two user preference models represented by two keyword vectors.

[0109] The Semantic Feature Refinement Component

[0110] It is not initially known whether a text paragraph in a document is relevant to a media object in the same document. In light of this, the media agent **210** may save many features surrounding the media object. These saved or extracted feature indices stored in the personal media database **226** may contain many redundant or non-related texts. Redundant text features decrease performance searches on the database **226**. Additionally, providing such redundant or non-related text in a search result may confuse a user. Hence the database **226** text features are refined either by deleting non-relevant features or by decreasing their importance or relevance weights. These aims are accomplished in view of the user action log **228**.

[0111] Each record in the user log **228** includes a user inserted text portion (e.g., user typed, spoken, or otherwise inserted), and an attached media object. In light of this, all text portions of the user log records **228** that contain the same media objects are combined together. Keyword frequencies across this combined text are calculated. These keyword frequency calculations replace the keyword weights in the original semantic features that correspond to the same media object.

[0112] Moreover, a number of top frequency keywords (e.g., the three (3) top frequency keywords) are selected along with their respective locations relative to the media object in the original document layout. Other media objects that came from similar sources (e.g., the same web page, the same website, and e-mail from the same person) are then examined. If a keyword in the semantic features of the other evaluated media came from the same relative locations of those top frequency keywords, there is confidence that the keyword's weight should be increased some amount (e.g., 10%). In this way, portions of original document layout that are more relevant to the semantics of the media object are determined.

[0113] An Exemplary Media Agent Process and Data Flow

[0114] **FIG. 3** shows exemplary aspects of process and data flows between modules and data sinks in the media agent module **210**. Specifically, **FIG. 3** shows sequences in which data transfer, use, and transformation are performed during the execution of the media agent **210** of **FIG. 2**. Data flow is illustrated with lines between respective modules and data sinks. Actions or events that trigger or cause execution and subsequent data flow are shown with lines capped with circles rather than arrows.

[0115] Sources of media content **300** include, for example, the WWW or the Web **302**, e-mail messages **304**, local and remote documents or Web pages **306**, local and remote file folders **308**, and so on. These media content sources **300** are only a few examples of the many possible sources of the media content pieces **106** of **FIG. 1**. On-line and offline crawler modules **212** and **214** retrieve semantic text description from these media content sources **300** to subsequently store into the personal media database **226**. Trigger **310** shows that the on-line crawler is activated by user actions **312** such as accessing the Web **302**, e-mails **304**, documents or Web pages **306**, file folders **308**, user "save-as" actions **316**, browsing **318**, text insertion **320**, media insertion **321**, and so on.

[0116] The offline crawler **214** is activated to access any number of these media sources **300** at system idle or as otherwise indicated by a user. The particular media sources **300** that are to be accessed by the offline crawler **214** are determined by information stored in the user preference models **230**. As discussed above, the learning module **222** and more specifically, the user preferences modeling submodule generates the user preference models **230** from records of user actions stored in the user actions log **228**. These data flows from the user action **312**, to the user action log **228**, to the learning module **222**, the user preference models **230**, and the offline crawler module **214** are shown by lines **314-1** through **314-4**. Note that the dataflow **314-4** is a trigger. This means that a user can implicitly or explicitly express a preference for when and where the offline crawler **214** is to obtain its information.

[0117] The user intention model **232** stores information that is used by the prediction module **216** to predict or anticipate user intention. As discussed above, the user intention model data store **232** is generated by learning module **222** and more specifically by user intention modeling submodule based on lexics, syntax, and/or patterns evaluated in training data such as data obtained from user action log **228**. The user action log **228** stores records of the user's actions. Each record includes a text portion and an indication of whether or not a media file is part of the action. Direct associations between the user actions, corresponding media content, and user intentions are determined on lexical, syntactical, and/or pattern basis. These direct associations are stored in the user intention model **232** data store. Arrows **314-1**, **323-1**, and **323-2** represent the data flow from the user action **312** to the user intention model data store **232**.

[0118] Certain user actions **312** cause media agent **210** modules to predict or anticipate user actions to provide semantically related suggestions to the user. Examples of such user actions **312** include, a "save as . . . " action **316**, a "browse . . . " action **318**, the insertion of text **320**, and an insert item action **322**. This action based trigger is illustrated by line **326-1**. Responsive to such user actions, prediction module **216** evaluates direct associations between the user actions, corresponding media content, and user intentions. (These direct associations are stored in the user intention model data store **232** as shown by dataflow line **323-3**). The prediction module **216** anticipates that the user desires to work with media files based on the user action **312** (e.g., the text information typed by the user, the selection of an insert media content menu item, and so on).

[0119] If the prediction module **216** determines that the user desires to work with media files, the prediction module **216** generates a potential search query vector (e.g., the query vector **234** of **FIG. 2**) from relevant information derived from evaluation of the user action **312** in view of the information in the user intention model **232**. This query vector may have been partially or wholly formulated from text typed in by the user for any number of reasons, including in response to an explicit user search for information. The prediction module **216**triggers and communicates the predicted query vector to the search engine **218**. This particular data flow is represented by arrows **326-2**, through **326-3**. Note that line **326-1** shows that user action **312** triggered the prediction module **216**. Note also that line **326-2** shows that the prediction module **216** triggered the search engine **218**.

[0120] The search engine **218** receives a search query vector (e.g., the query vector **234** of **FIG. 2**) that may have been wholly or partially generated from information (e.g., text) input by the user or wholly or partially generated by the prediction module **216**. The search engine **218** evaluates this query in view of the semantic media content text indices stored in the personal media database **226**. (Recall that these indices are created by the online and offline crawlers **212** and **214**. These indices are also created by the feature refinement sub-module of the learning module **222** as described below).

[0121] If a relevant set of media objects in the personal media database **226** are identified, the search engine **218** triggers and communicates the identified media objects to the suggestion module **220**. This information may be sorted and communicated to the suggestion module **220** according to each items semantic similarity to the search query. These data flow are represented by lines **326-1** through **326-4**. Note that lines **326-1**, **326-2**, and **326-4** are triggers.

[0122] The suggestion module **220** receives a list of relevant media content from the search engine **218**. This information is displayed to a user for viewing and response (e.g., selection or editing of a suggestion, editing of the semantic text corresponding to the suggestion, cancellation of the task, and so on). These data flow are represented by lines **326-4** through **326-5**. Note that line **326-4** is a trigger.

[0123] The learning module **222** and specifically the feature refinement sub-module, refines the semantic media content text indices stored in the personal media database **226**. To accomplish this, the feature refinement sub-module evaluates text and corresponding media content in the user action log **228** to evaluate corresponding keyword frequencies or keyword relevance. The feature refinement sub-module uses this keyword evaluation to redefine keyword weights in the personal media database **226** to correlate with a compiled history of user actions. This redefinition of keyword weights may result in removal of certain **226** keywords in indices in the database that are determined to be semantically non-related at that point in time. These data flow are represented by lines **328-1** and **328-2**.

[0124] An Exemplary Media Agent Procedure

[0125] **FIG. 4** shows an exemplary procedure **400** to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 4** shows a procedure **400** for a media agent **210** of **FIGS. 2 and 3** to determine whether offline gathering of media content semantics, online gathering of media content semantics, preference and intention modeling, or user intention prediction and suggestion procedures should be performed.

[0126] At block **402**, the procedure determines if a user action (e.g., a mouse move, a key press, saving of a file, downloading a file, following a link or hyperlink on a network, and so on) is received. If not, the procedure continues at block **404**, wherein it is determined if the system **102** of the media agent **210** is idle for some reason (e.g., a pause between keystrokes, etc.). A system is in an idle state when it is operational and in service but one or more processing cycles is still available for use. Having determined that the system is not in an idle state, the procedure continues at block **402**, as described above.

[0127] If the system is idle (block **404**), the procedure **400** continues at block **406**, wherein the offline crawler program module **214** extracts semantic text features from media content sources (e.g., e-mails, documents, memory caches, etc.), if any, according to the user preference model **230**. The user preference model **230** indicates learned aspects of a user's behavior with respect to media content locations, preferred or frequently accessed media content, and so on. These learned aspects identify preferred media content and respective semantic features to extract and store while the system is idle.

[0128] At block **408**, the procedure stores any extracted semantic features and corresponding media content (block **404**) into the user's personalized media database **226**. In this manner the database **226** includes personalized semantic indices (PSI) that reflect all of the media content accessed by the user. Since each user may have different preferences of his/her favorite media objects, the personal index may differ from user to user. The procedure continues at block **402**.

[0129] At block **402**, responsive to receiving a user action, the procedure continues at block **410**, wherein a user action log (i.e., log **228** of **FIGS. 2 and 3**) is updated to include a record of the text and/or media content corresponding to the user's action (block **402**). At block **412**, it is determined if the action corresponds to user access of a URL, opening of a media file, or downloading a file (e.g., saving a file). If so, the procedure continues at on-page reference "B", as shown in **FIG. 5**.

[0130] **FIG. 5** shows further aspects of an exemplary procedure **400** to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 5** shows further aspects of

a procedure for a media agent **210** of **FIGS. 2 and 3** to perform online gathering of media content semantics and preference and intention modeling. Reference "B" indicates that procedure **400** executes blocks **502** through **510**. Although the blocks are orderly numbered, the ordering does not imply any preferred sequence of execution. For instance, blocks **502** and **504** may be executed before blocks **506** through **510**, vice versa, and so on.

[0131] At block **502**, the procedure **400** (i.e., the online crawler **212** of **FIGS. 2 and 3**) extracts semantic media content features (i.e., text features) from the media content itself and/or from a document (e.g., e-mail, Web page, etc.) corresponding to the media content. Recall that this operation (block **502**) is performed responsive to a user action (e.g., a URL access, an open file action, a save as action, and so on). At block **504**, the extracted semantic features and corresponding media content are stored in the user's personal media database **226**. It can be appreciated that the semantic features can be stored separately, if desired, from the media content.

[0132] At block **506**, user preference modeling is performed. As discussed above, the learning module **222** of **FIGS. 2 and 3** and more specifically, the user preferences modeling sub-module (see, **FIG. 3**) generates the user preference models **230** from records of user actions stored in the user actions log **228**.

[0133] At block **508**, the procedure **400** performs user intention modeling to store information that is used by the prediction module **216** of **FIGS. 2 and 3** to predict or anticipate user intention. As discussed above, the user intention model data store **232** is generated by learning module **222** and more specifically by user intention modeling sub-module of **FIG. 3** based on lexics, syntax, and/or patterns evaluated in training data such as data obtained from user action log **228**.

[0134] At block **510**, the procedure **400** refines the semantic features corresponding to media content stored in the personal media database **226** of **FIGS. 2 and 3**. The feature refinement sub-module of **FIG. 3** performs this operation by evaluating text features and corresponding media content in the user action log **228** to evaluate corresponding keyword frequencies or keyword relevance. The feature refinement sub-module uses this keyword evaluation to redefine or update keyword weights in the personal media database **226** to correlate or with a compiled history of user actions. At this point, the procedure **400** continues at block **402**, as shown by the on-page reference "A" of **FIG. 4**.

[0135] Recall that at block **412** of **FIG. 4**, the procedure **400** determines if the identified user action (block **402**) is of a particular type of user action (e.g., URL access, media file open, media file save as, and so on). If so, the discussed procedures of **FIG. 5** were performed. However, if the user action was not of the particular type, the procedure **400** continues at block **602** of **FIG. 6**, as illustrated by on-page reference "C".

[0136] **FIG. 6** shows further aspects of exemplary procedures to automatically collect, manage, and suggest information corresponding to personalized use of media content. More specifically, **FIG. 6** shows further aspects of a procedure for a media agent of **FIGS. 2 and 3** to determine whether preference and intention modeling or user intention prediction and suggestion procedures should be performed. At block **602**, the procedure **400** determines if the user action (block **402**) is an explicit user search for a media object, an object insertion action, or action corresponding to a document edit (e.g., e-mail, a word-processing document, etc.). If not, the procedure continues at block **402**, as indicated by the on-page reference "A" of page 4.

[0137] Otherwise, at block **604**, the procedure (i.e., the prediction module **216** of **FIGS. 2 and 3**) generates a set of media content predictions using a user intention model **232** of **FIGS. 2 and 3**. A search query vector (e.g., the query vector **234** of **FIG. 2**) is generated from the media content predictions in view of the user action (block **602**). At block **606**, the procedure **400** uses the generated query vector (block **604**) to search the user's personal media database **226** of **FIGS. 2 and 3** for corresponding media content. At block **608**, identified media content information (e.g., file names, URLs, etc . . .) is displayed or "suggested" to the user for subsequent evaluation, selection, and/or other response (e.g., editing). The procedure continues at block **402**, as indicated by the on-page reference "A" of page 4.

[0138] An Exemplary Suitable Computing Environment

[0139] **FIG. 7** illustrates aspects of an exemplary suitable operating environment in which a media agent to semantically index, suggest, and retrieve media content information according to personal usage patterns may be implemented. The illustrated operating environment is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Other well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics (e.g., digital video recorders), gaming consoles, cellular telephones, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0140] **FIG. 7** shows a general example of a computer **742** that can be used in accordance with the described arrangements and procedures. Computer **742** is shown as an example of a computer in which various embodiments of the invention can be practiced, and can be used to implement, for example, a client **102** of **FIG. 1**, a media agent **210**, online and offline crawler components **212** and **214**, prediction component **216**, search engine component **218**, suggestion component **220**, or a learning component **222** of **FIGS. 2 and 3**, and so on. Computer **742** includes one or more processors or processing units **744**, a system memory **746**, and a bus **748** that couples various system components including the system memory **746** to processors **744**.

[0141] The bus **748** represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory **746** includes read only memory (ROM) **750** and random access memory (RAM) **752**. A basic input/output system (BIOS) **754**, containing the basic routines that help to transfer information between elements within computer **742**, such as during start-up, is stored in ROM **750**. Computer **742** further

includes a hard disk drive **756** for reading from and writing to a hard disk, not shown, connected to bus **748** via a hard disk drive interface **757** (e.g., a SCSI, ATA, or other type of interface); a magnetic disk drive **758** for reading from and writing to a removable magnetic disk **760**, connected to bus **748** via a magnetic disk drive interface **761**; and an optical disk drive **762** for reading from and/or writing to a removable optical disk **764** such as a CD ROM, DVD, or other optical media, connected to bus **748** via an optical drive interface **765**. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for computer **742**. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk **760** and a removable optical disk **764**, it will be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, random access memories (RAMs), read only memories (ROM), and the like, may also be used in the exemplary operating environment.

[0142] A number of program modules may be stored on the hard disk, magnetic disk **760**, optical disk **764**, ROM **750**, or RAM **752**, including an operating system **770**, one or more application programs **772**, other program modules **774**, and program data **776**. A user may enter commands and information into computer **742** through input devices such as keyboard **778** and pointing device **780**. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit **744** through an interface **768** that is coupled to the system bus (e.g., a serial port interface, a parallel port interface, a universal serial bus (USB) interface, etc.). A monitor **784** or other type of display device is also connected to the system bus **748** via an interface, such as a video adapter **786**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

[0143] Computer **742** operates in a networked environment using logical connections to one or more remote computers, such as a remote computer **788**. The remote computer **788** may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer **742**, although only a memory storage device **790** has been illustrated in **FIG. 7**. The logical connections depicted in **FIG. 7** include a local area network (LAN) **792** and a wide area network (WAN) **794**. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet. In certain embodiments of the invention, computer **742** executes an Internet Web browser program (which may optionally be integrated into the operating system **770**) such as the "Internet Explorer" Web browser manufactured and distributed by Microsoft Corporation of Redmond, Washington.

[0144] When used in a LAN networking environment, computer **742** is connected to the local network **792** through a network interface or adapter **796**. When used in a WAN networking environment, computer **742** typically includes a modem **798** or other means for establishing communications over the wide area network **794**, such as the Internet. The modem **798**, which may be internal or external, is connected to the system bus **748** via a serial port interface **768**. In a networked environment, program modules depicted relative to the personal computer **742**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0145] Computer **742** also includes a broadcast tuner **799**. Broadcast tuner **799** receives broadcast signals either directly (e.g., analog or digital cable transmissions fed directly into tuner **799**) or via a reception device (e.g., via antenna or satellite dish).

[0146] Computer **742** typically includes at least some form of computer readable media. Computer readable media can be any available media that can be accessed by computer **742**. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data.

[0147] Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other media which can be used to store the desired information and which can be accessed by computer **742**. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0148] The invention has been described in part in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0149] For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

[0150] Alternatively, the invention may be implemented in hardware or a combination of hardware, software, and/or firmware. For example, one or more application specific integrated circuits (ASICs) could be designed or programmed to carry out the invention.

[0151] Conclusion

[0152] Although the description above uses language that is specific to structural features and/or methodological acts, it is to be understood that the described arrangements and procedures defined in the appended claims are not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the described arrangements and procedures.

1. In a computer system, a method comprising:

detecting user input;

analyzing the user input;

predicting desired access to one or more media files based on the analysis;

retrieving information corresponding to one or more media files from a media content source; and

presenting the information to a user for suggested access.

2. A method as recited in claim 1, wherein the user input is text.

3. A method as recited in claim 1, wherein the user input is text in a word processor document or in an e-mail.

4. A method as recited in claim 1, wherein the information further comprises suggested media content items, the method further comprising:

detecting user interest in an item of the suggested media items; and

responsive to detecting the user interest, displaying a high-level feature corresponding to the item, the high-level feature being stored in a database.

5. A method as recited in claim 1, wherein analyzing the user input further comprises determining one or more keywords from the text, and wherein the one or more media files correspond to the one or more keywords.

6. A method as recited in claim 1, wherein analyzing the user input further comprises evaluating the user input based on lexical features.

7. A method as recited in claim 1, wherein analyzing the user input further comprises evaluating the user input based on syntactical features.

8. A method as recited in claim 1, wherein analyzing the user input further comprises evaluating the user input based on at least a partially instantiated sentence pattern.

9. A method as recited in claim 1, wherein the method further comprises identifying media content use patterns, and wherein analyzing the user input further comprises evaluating the user input based on the media content use patterns.

10. In a computer system, a method comprising:

detecting user access of a media content source;

responsive to detecting the user access:

collecting a piece of media content and associated text from the media content source;

extracting semantic text features from the associated text and the piece of media content; and

indexing the semantic text features into a media database.

11. A method as recited in claim 10, wherein the media database is a personalized media database.

12. A method as recited in claim 10, further comprising indexing the piece of media content in the media database or separate from the media database.

13. A method as recited in claim 10, wherein the media content source comprises an e-mail message, wherein the media content is an attached to the e-mail message or a link to the media content, and wherein the associated text is a body of the e-mail message.

14. A method as recited in claim 10, wherein the media content source comprises a word processing document, wherein the media content is inserted media content, and wherein the associated text is document text.

15. A method as recited in claim 10, wherein the media content source comprises a Web page.

16. A method as recited in claim 10, wherein at least a portion of the semantic text features in the media database include an indication of relevancy with respect to individual ones of the media content, and wherein the method further comprises:

monitoring user actions;

identifying patterns of media content use from the user actions; and

for each semantic text feature in the at least a portion, modifying its indication of relevancy to individual ones of the media content based on the patterns of media content use.

17. A method as recited in claim 10, further comprising:

detecting user input;

performing an analysis of the user input to determine that the user desires to access media content;

responsive to performing the analysis:

generating search criteria based on linguistic features of the user input;

identifying, based at least in part on the search criteria, one or more media files that are semantically related to the user input from the media database; and

presenting information corresponding to the one or more media files to the user.

18. A method as recited in claim 17, wherein the linguistic features are lexical, syntactical, and/or partial sentence pattern features.

19. A method as recited in claim 17, wherein the analysis is based at least in part on patterns of previous media content use, the patterns corresponding to the user.

20. In a computer system, a method comprising:

monitoring a plurality of user actions;

determining media content use preferences based on the user actions;

collecting media content and associated text from a media content source;

extracting semantic text features from the media content and the associated text;

determining that the media content is of interest to the user based at least in part on semantic similarity between the media content use preferences and the semantic text features; and

responsive to determining that the media content is of interest to the user, indexing the semantic text features into a media database.

21. A method as recited in claim 19, wherein the media content use preferences are further based on keywords extracted from information corresponding to the user actions.

22. A method as recited in claim 19, wherein the media content use preferences comprise a plurality of user preference models, each user preference model comprising semantically similar keywords that correspond to the user actions, and wherein media content is determined to be of interest to the user if there is semantic similarity between the media content and at least one of the user preference models.

23. A method as recited in claim 19, further comprising:

detecting that the computer system is in an idle state; and

wherein the acts of collecting, extracting and determining are performed responsive to detecting the idle state.

24. A computer-readable medium comprising computer-executable instructions for:

detecting user input;

responsive to detecting the user input:

analyzing the user input;

predicting desired access to one or more media files based on the analysis;

retrieving information corresponding to one or more media files from a media content source; and

presenting the information as a suggestion.

25. A computer-readable medium as recited in claim 24, wherein the user input is text.

26. A computer-readable medium as recited in claim 24, wherein the user input corresponds to an e-mail message or a word processing document.

27. A computer-readable medium as recited in claim 24, wherein the information further comprises suggested media content items, and wherein the computer-executable instructions further comprise instructions for:

detecting user interest in an item of the suggested media items; and

responsive to detecting the user interest, displaying a high-level feature corresponding to the item, the high-level feature being stored in a database.

28. A computer-readable medium as recited in claim 24, wherein the instructions for analyzing the user input further comprise determining one or more keywords from the user input, and wherein the one or more media files correspond to the one or more keywords.

29. A computer-readable medium as recited in claim 24, wherein the instructions for analyzing the user input further comprise evaluating the user input based on lexical features.

30. A computer-readable medium as recited in claim 24, wherein the instructions for analyzing the user input further comprise evaluating the user input based on syntactical features.

31. A computer-readable medium as recited in claim 24, wherein the instructions for analyzing the user input further comprise evaluating the user input based on at least a partially instantiated sentence pattern.

32. A computer-readable medium as recited in claim 24, wherein the computer-executable instructions further comprise instruction for identifying media content use patterns, and wherein analyzing the user input further comprises evaluating the user input based on the media content use patterns.

33. A computer-readable medium comprising computer-executable instructions for:

detecting user access of a media content source;

responsive to detecting the user access:

collecting a piece of media content and associated text from the media content source;

extracting semantic text features from the associated text and the piece of media content; and

indexing the semantic text features into a media database.

34. A computer-readable medium as recited in claim 33, further comprising computer-executable instructions for indexing the piece of media content in the media database or separate from the media database.

35. A computer-readable medium as recited in claim 33, wherein the media content source is an e-mail message, wherein the media content is an attached to the e-mail message or a link to the media content, and wherein the associated text is a body of the e-mail message.

36. A computer-readable medium as recited in claim 33, wherein the media content source comprises a word processing document, wherein the media content is inserted media content, and wherein the associated text is document text.

37. A computer-readable medium as recited in claim 33, wherein the media content source comprises a Web page.

38. A computer-readable medium as recited in claim 33, wherein at least a portion of the semantic text features in the media database include an indication of relevancy with respect to individual ones of the media content, and wherein the computer-executable instructions further comprise instructions for:

monitoring user actions;

identifying patterns of media content use from the user actions; and

for each semantic text feature in the at least a portion, modifying its indication of relevancy to individual ones of the media content based on the patterns of media content use.

39. A computer-readable medium as recited in claim 33, further comprising instructions for:

detecting an action by a user comprising insertion of text;

performing an analysis of the text to determine that the user desires to access media content;

responsive to performing the analysis:

generating search criteria based on linguistic features of the text;

identifying, based at least in part on the search criteria, one or more media files that are semantically related to the text from the media database; and

presenting information corresponding to the one or more media files to the user.

40. A computer-readable medium as recited in claim 39, wherein the linguistic features are lexical, syntactical, and/or partial sentence pattern features.

41. A computer-readable medium as recited in claim 39, wherein the analysis is based at least in part on patterns of previous media content use, the patterns corresponding to the user.

42. A computer-readable medium comprising computer-executable instructions for:

monitoring a plurality of user actions;

determining media content use preferences based on the user actions;

collecting media content and associated text from a media content source;

extracting semantic text features from the media content and the associated text;

determining that the media content is of interest to the user based at least in part on semantic similarity between the media content use preferences and the semantic text features; and

responsive to determining that the media content is of interest to the user, indexing the semantic text features into a media database.

43. A computer-readable medium as recited in claim 42, wherein the media content use preferences are further based on keywords extracted from information corresponding to the user actions.

44. A computer-readable medium as recited in claim 42, wherein the media content use preferences comprise a plurality of user preference models, each user preference model comprising semantically similar keywords that correspond to the user actions, and wherein media content is determined to be of interest to the user if there is semantic similarity between the media content and at least one of the user preference models.

45. A computer-readable medium as recited in claim 42, further comprising computer-executable instructions for:

detecting that the computer system is in an idle state; and

wherein the instructions for collecting, extracting and determining are performed responsive to detecting the idle state.

46. A computing device comprising:

a processor:

a memory coupled to the processor, the memory comprising computer-executable instructions, the processor being configured to fetch and execute the computer-executable instructions for:

detecting user input;

analyzing the user input;

predicting desired access to one or more media files based on the analysis;

retrieving information corresponding to one or more media files from a media content source; and

presenting the information as a suggestion.

47. A computing device as recited in claim 46, wherein the user input comprises insertion of text into a document such as an e-mail message or word processing document.

48. A computing device as recited in claim 46, wherein the information further comprises suggested media content items, and wherein the computer-executable instructions further comprise:

detecting user interest in an item of the suggested media items; and

responsive to detecting the user interest, displaying a high-level feature corresponding to the item, the high-level feature being stored in a database.

49. A computing device as recited in claim 46, wherein the instructions for analyzing the user input further comprise instructions for determining one or more keywords from the user input, and wherein the one or more media files correspond to the one or more keywords.

50. A computing device as recited in claim 46, wherein the instructions for analyzing the user input further comprise evaluating the user input based on lexical features.

51. A computing device as recited in claim 46, wherein the instructions for analyzing the user input further comprise evaluating the user input based on syntactical features.

52. A computing device as recited in claim 46, wherein the instructions for analyzing the user input further comprise evaluating the user input based on at least a partially instantiated sentence pattern.

53. A computing device as recited in claim 46, wherein the computer-executable instructions further comprise instruction for identifying media content use patterns, and wherein analyzing the user input further comprises evaluating the user input based on the media content use patterns.

54. A computing device comprising:

processing means for:

detecting user input;

analyzing the user input;

predicting desired access to one or more media files based on the analysis;

retrieving information corresponding to one or more media files from a media content source; and

presenting the information as a suggestion.

55. A computing device comprising:

a processor:

a memory coupled to the processor, the memory comprising computer-executable instructions, the processor being configured to fetch and execute the computer-executable instructions for:

detecting user access of a media content source;

responsive to detecting the user access:

collecting a piece of media content and associated text from the media content source;

extracting semantic text features from the associated text and the piece of media content; and

indexing the semantic text features into a media database.

56. A computing device as recited in claim 55, further comprising computer-executable instructions for indexing the piece of media content in the media database or separate from the media database.

57. A computing device as recited in claim 55, wherein the media content source is an e-mail message, wherein the media content is an attached to the e-mail message or a link to the media content, and wherein the associated text is a body of the e-mail message.

58. A computing device as recited in claim 55, wherein the media content source comprises a word processing document, wherein the media content is inserted media content, and wherein the associated text is document text.

59. A computing device as recited in claim 55, wherein the media content source comprises a Web page.

60. A computing device as recited in claim 55, wherein at least a portion of the semantic text features in the media database include an indication of relevancy with respect to individual ones of the media content, and wherein the computer-executable instructions further comprise instructions for:

monitoring user actions;

identifying patterns of media content use from the user actions; and

for each semantic text feature in the at least a portion, modifying its indication of relevancy to individual ones of the media content based on the patterns of media content use.

61. A computing device as recited in claim 55, further comprising instructions for:

detecting an action by a user comprising insertion of text;

performing an analysis of the text to determine that the user desires to access media content;

responsive to performing the analysis:

generating search criteria based on linguistic features of the text;

identifying, based at least in part on the search criteria, one or more media files that are semantically related to the text from the media database; and

presenting information corresponding to the one or more media files to the user.

62. A computing device as recited in claim 61, wherein the linguistic features are lexical, syntactical, and/or partial sentence pattern features.

63. A computing device as recited in claim 61, wherein the analysis is based at least in part on patterns of previous media content use, the patterns corresponding to the user.

64. A computing device comprising:

processing means for:

detecting user access of a media content source;

responsive to detecting the user access:

collecting a piece of media content and associated text from the media content source;

extracting semantic text features from the associated text and the piece of media content; and

indexing the semantic text features into a media database.

65. A computing device comprising:

a processor:

a memory coupled to the processor, the memory comprising computer-executable instructions, the processor being configured to fetch and execute the computer-executable instructions for:

monitoring a plurality of user actions;

determining media content use preferences based on the user actions;

collecting media content and associated text from a media content source;

extracting semantic text features from the media content and the associated text;

determining that the media content is of interest to the user based at least in part on semantic similarity between the media content use preferences and the semantic text features; and

responsive to determining that the media content is of interest to the user, indexing the semantic text features with the media content into a media database.

66. A computing device as recited in claim 65, wherein the media database is a personalized media database.

67. A computing device as recited in claim 65, wherein the media content use preferences are further based on keywords extracted from information corresponding to the user actions.

68. A computing device as recited in claim 65, wherein the media content use preferences comprise a plurality of user preference models, each user preference model comprising semantically similar keywords that correspond to the user actions, and wherein media content is determined to be of interest to the user if there is semantic similarity between the media content and at least one of the user preference models.

69. A computing device as recited in claim 65, further comprising computer-executable instructions for:

detecting that the processor is in an idle state; and

wherein the instructions for collecting, extracting and determining are performed responsive to detecting the idle state.

70. A computing device comprising:

processing means for:

monitoring a plurality of user actions;

determining media content use preferences based on the user actions;

collecting media content and associated text from a media content source;

extracting semantic text features from the media content and the associated text;

determining that the media content is of interest to the user based at least in part on semantic similarity between the media content use preferences and the semantic text features; and

responsive to determining that the media content is of interest to the user, indexing the semantic text features into a media database.

71. A computing device as recited in claim 70, further comprising processing means for:

detecting that the processor is in an idle state; and

wherein the means for collecting, extracting and determining are performed responsive to detecting the idle state.

72. A method comprising:

determining that a user wants to save or download a media object from a media source;

extracting semantic information from the media source; and

suggesting a filename to the user for the media object based on the semantic information.

73. A method as recited in claim 72, wherein the media source comprises a Web page.

74. A method as recited in claim 72, wherein the media source comprises an e-mail message.

75. A method as recited in claim 72, wherein the filename is selectable and editable.

76. A method as recited in claim 72, wherein the semantic information is based on any combination of one or more of a filename, text, a title, a keyword, or a hyperlink extracted from the media content or the media source.

77. A device comprising:

processing means for:

determining that a user wants to save or download a media object from a media source;

extracting semantic information from the media source; and

suggesting a filename to the user for the media object based on the semantic information.

78. A device as recited in claim 77, wherein the media source comprises a Web page.

79. A device as recited in claim 77, wherein the media source comprises an e-mail message.

80. A device as recited in claim 77, wherein the filename is selectable and editable.

81. A device as recited in claim 77, wherein the semantic information is based on any combination of one or more of a filename, text, a title, a keyword, or a hyperlink extracted from the media content or the media source.

82. A computer-readable medium comprising computer-executable instructions for:

determining that a user wants to save or download a media object from a media source;

extracting semantic information from the media source; and

suggesting a filename to the user for the media object based on the semantic information.

83. A computer-readable medium as recited in claim 82, wherein the media source comprises a Web page.

84. A computer-readable medium as recited in claim 82, wherein the media source comprises an e-mail message.

85. A computer-readable medium as recited in claim 82, wherein the filename is selectable and/or editable.

86. A computer-readable medium as recited in claim 82, wherein the semantic information is based on any combination of one or more of a filename, text, a title, a keyword, or a hyperlink extracted from the media content or the media source.

* * * * *